EECE 5639 Computer Vision I

Lecture 22

- **SFM; Object Recognition**
- **Project 4 is out. Now Due April 16.**
- **Hw 5 is out. Now Due April 19**

Next Class

- **Object Recognition**

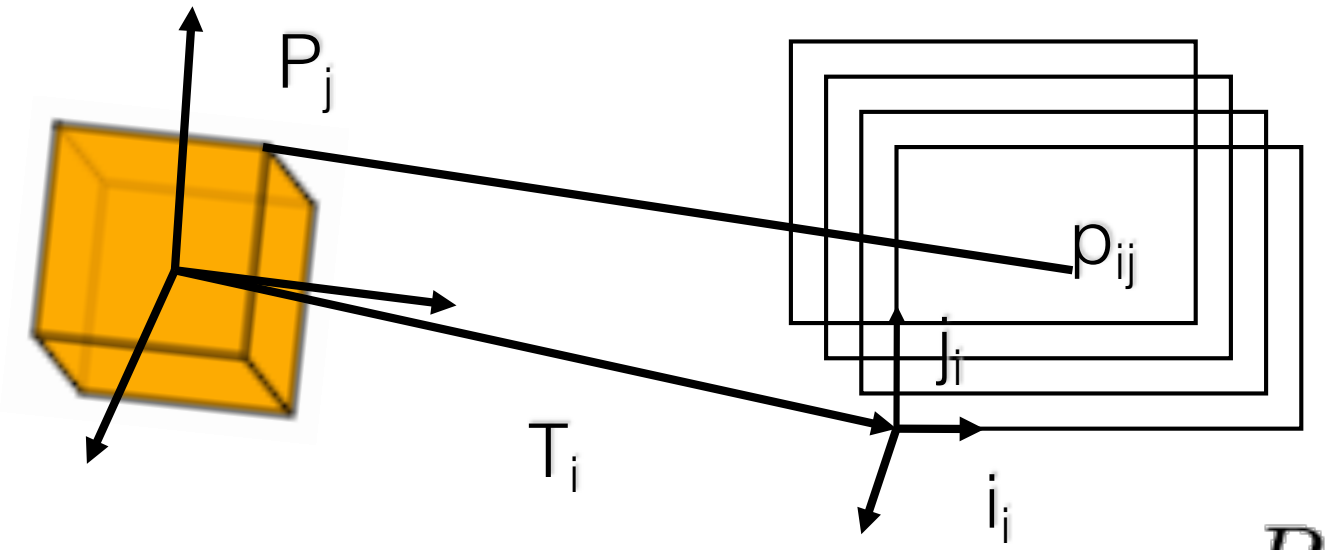
SFM by Factorization

Consider a FIXED camera and a MOVING object

Track n features across m frames.

We want to find the structure and the motion of the object.

Orthographic Projection



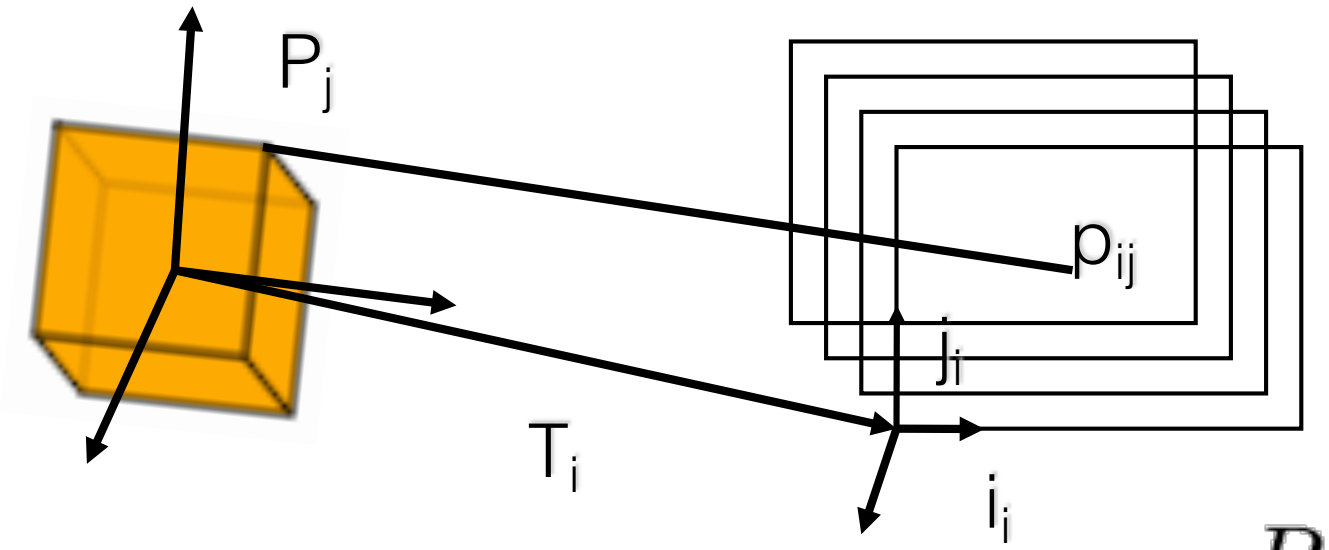
$$p_{ij} = \underset{2x3}{R_j} \underset{3x1}{P_j} + \underset{2x1}{T_i}$$

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} = \begin{bmatrix} R_1 & T_1 \\ R_2 & T_1 \\ \vdots & \vdots \\ R_m & T_m \end{bmatrix} \begin{bmatrix} P_1 & P_2 & \cdots & P_n \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

W M S

$$W = MS$$

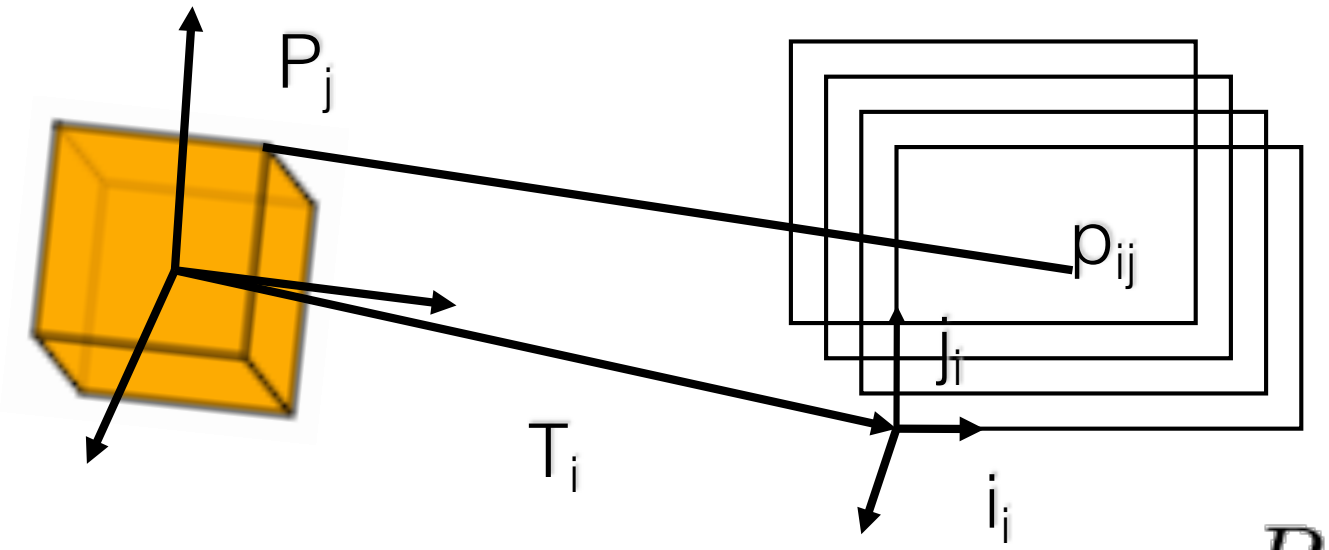
Orthographic Projection



$$p_{ij} = \underset{2x3}{R_j} \underset{3x1}{P_j} + \underset{2x1}{T_i}$$

One can eliminate T by setting the origin of both coordinate systems (the object and the image coordinate systems) at the centroids of all the points (3D and each 2D frame).

Orthographic Projection

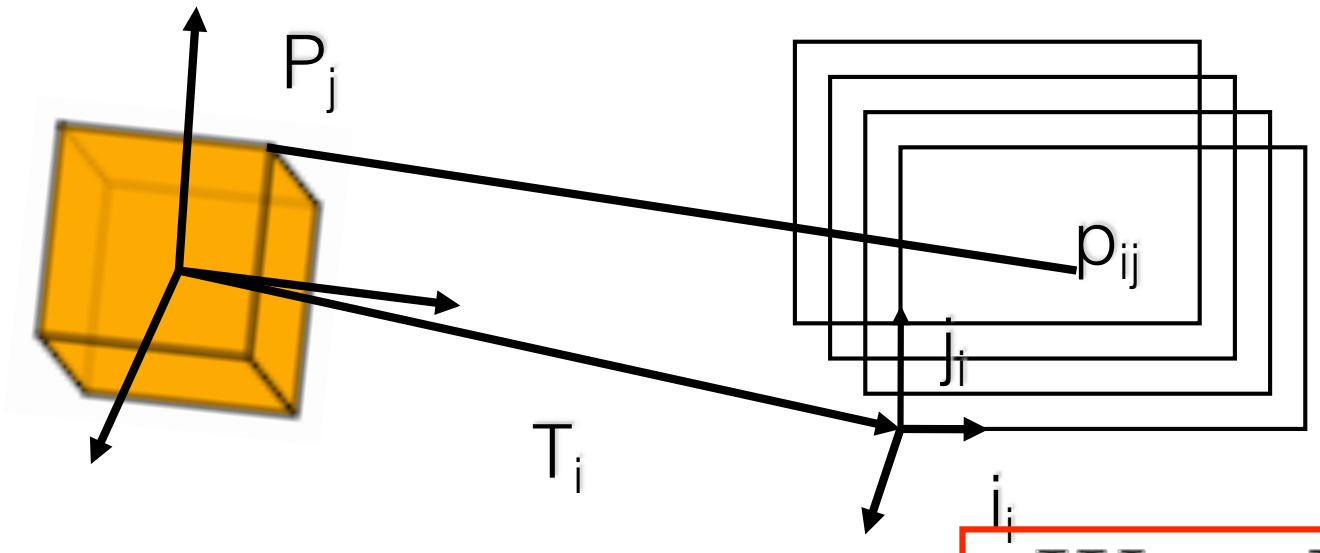


$$p_{ij} = \underset{2x3}{R_j} \underset{3x1}{P_j} + \underset{2x1}{T_i}$$

$$\begin{bmatrix} \hat{p}_{11} & \hat{p}_{12} & \cdots & \hat{p}_{1n} \\ \hat{p}_{21} & \hat{p}_{22} & \cdots & \hat{p}_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ \hat{p}_{m1} & \hat{p}_{m2} & \cdots & \hat{p}_{mn} \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_m \end{bmatrix} \begin{bmatrix} \hat{P}_1 & \hat{P}_2 & \cdots & \hat{P}_n \end{bmatrix}$$

W M S

$$W = MS$$



$$W = MS$$

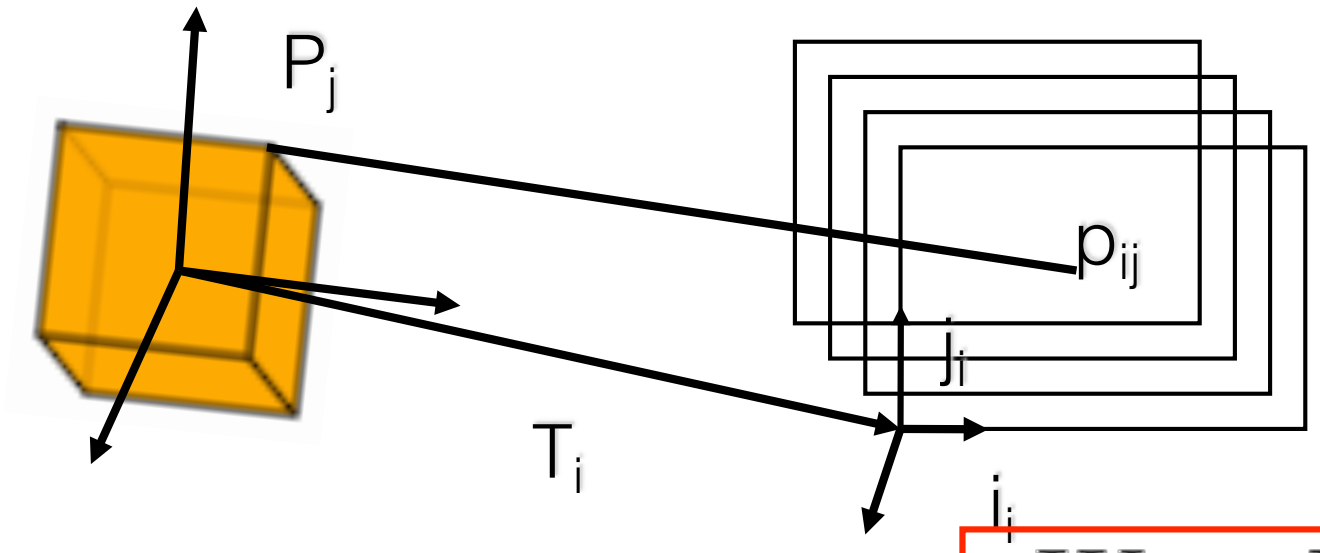
2m x n 2m x 3 3 x n

W is **rank 3** and M and S can be found (up to an affine transf.) from the SVD of W

$$W = UDV^T$$

$$M = UD^{\frac{1}{2}}Q \qquad S = Q^{-1}D^{\frac{1}{2}}V^T$$

Finding Q



$$W = MS$$

2m x n 2m x 3 3 x n

$$W = UDV^T \qquad S = Q^{-1}D^{\frac{1}{2}}V^T$$

$$M = UD^{\frac{1}{2}}Q$$

M should be a stack of ROTATIONS!

Finding Q

$$M = U D^{\frac{1}{2}} Q$$

M should be a stack of ROTATIONS!

$$\begin{aligned} m_{i1}.m_{i2} &= 0 \\ |m_{i1}|^2 &= 1 \\ |m_{i2}|^2 &= 1 \end{aligned}$$

$$m_{i1} \times m_{i2} = m_{i3}$$

Recovering T

$$T_i = R_i \begin{bmatrix} \bar{p}_i \\ \alpha \end{bmatrix} \quad \text{with } \alpha \text{ an arbitrary real number}$$

Algorithm Summary

Obtain the registered measurement matrix W (subtract the mean in each frame)

Compute SVD of $W=UDV^T$

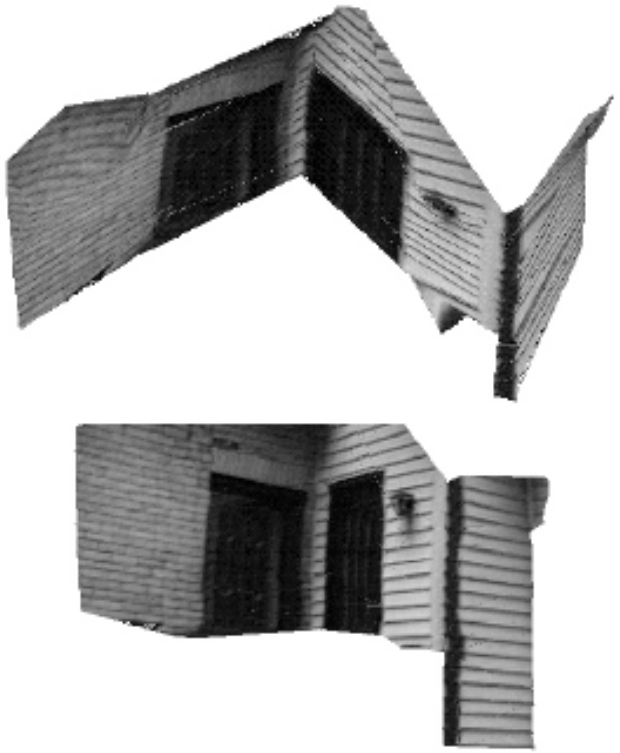
Keep the first 3 eigenvalues/eigenvectors: $W' = U'D'V'^T$

Define $M = U'D'^{1/2}$ and $S = D'^{1/2}V'^T$

Find affinity Q such that MQ is a stack of rotations

Compute T

Reconstruction Results (Tomasi and Kanade, 1992)



Reprinted from Tomasi and Kanade 1992

Further Factorization work

Factorization with uncertainty

Factorization for indep. moving objects (Irani & Anandan, IJCV'02)

Factorization for dynamic objects

Perspective factorization (Costeira and Kanade '94)

Factorization with outliers and missing pts. (Bregler et al. 2000, Brand 2001)

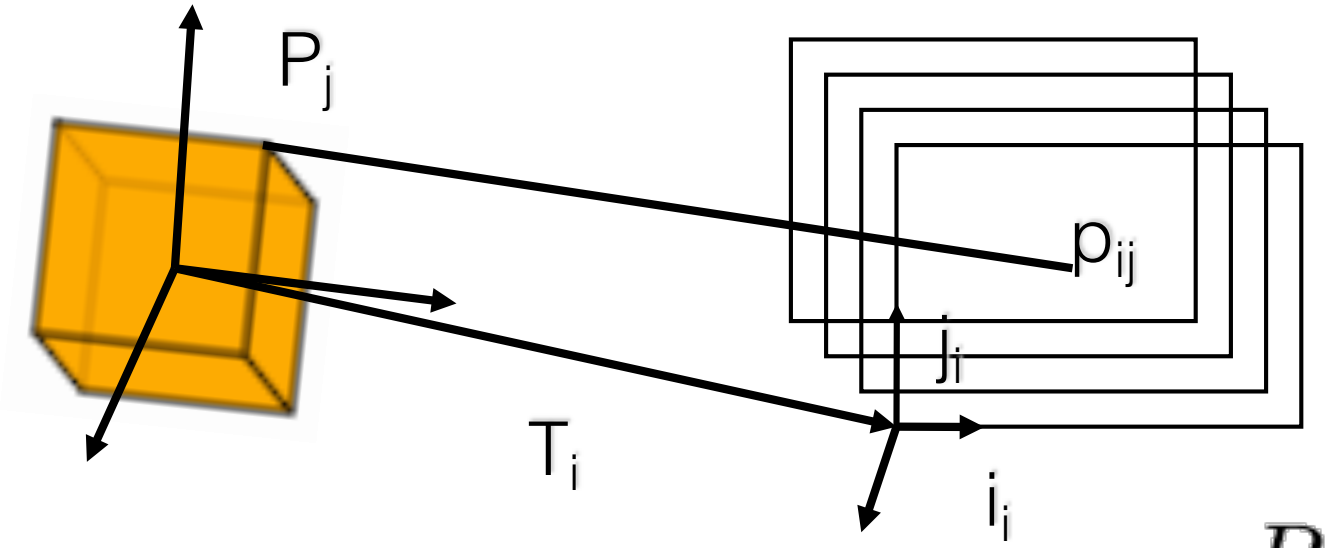
(Sturm & Triggs 1996, Ayazoglu, Sznaier, Camps 2010…)

(Jacobs 1997 (affine),
Martinek and Pajdla 2001,
Aanaes 2002 (perspective))

Motion Based Segmentation by Factorization

Costeira & Kanade ‘98

Motion based Segmentation by Factorization

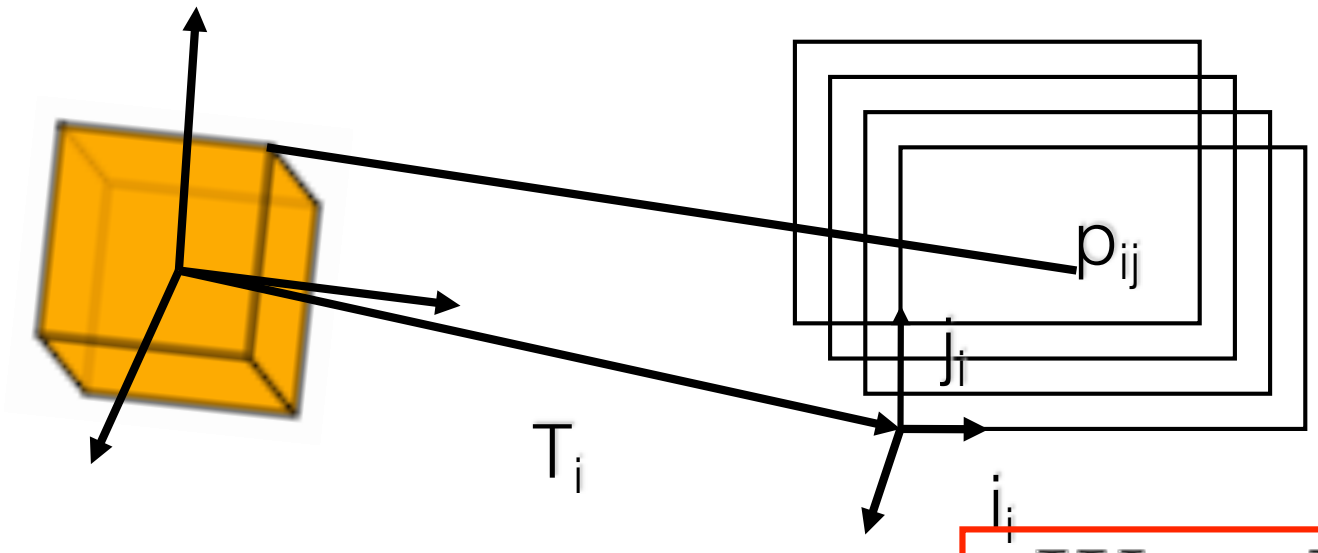


$$p_{ij} = R_i P_j + T_i$$

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} = \begin{bmatrix} R_1 & T_1 \\ R_2 & T_1 \\ \vdots & \vdots \\ R_m & T_m \end{bmatrix} \begin{bmatrix} P_1 & P_2 & \cdots & P_n \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

W M S

$$W = MS$$



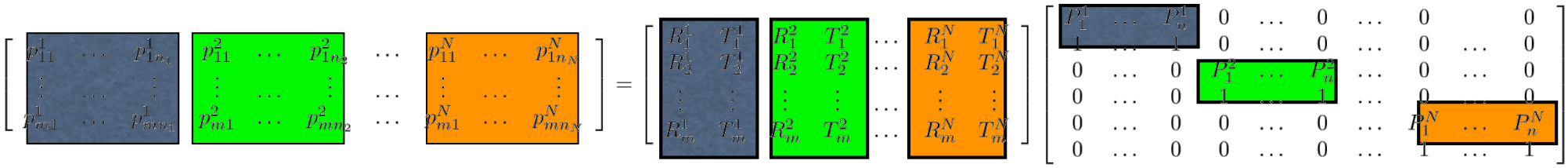
$$W = MS$$

2m x n 2m x 4 4 x n

W has at most rank 4

Motion based Segmentation by Factorization

For N (segmented) rigid objects tracked across m frames:



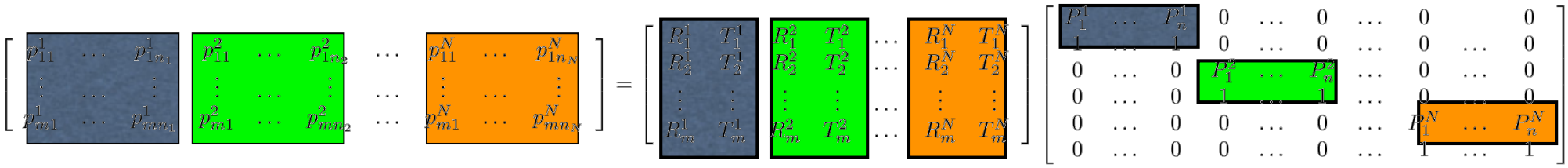
$2m \times (n_1+n_2+\ldots+n_N)$ $\quad 2m \times 4N \quad$ $4N \times (n_1+n_2+\ldots+n_N)$

$$W = MS$$

W has at most rank 4N

Motion based Segmentation by Factorization

For N (segmented) rigid objects tracked across m frames:



$2m \times (n_1+n_2+\ldots+n_N)$ $\quad 2m \times 4N$ $\quad 4N \times (n_1+n_2+\ldots+n_N)$

$$W = MS$$

$$W = UDV^T$$

$$Z = V(V^TV)^{-1}V^T = VV^T$$

Is block diagonal

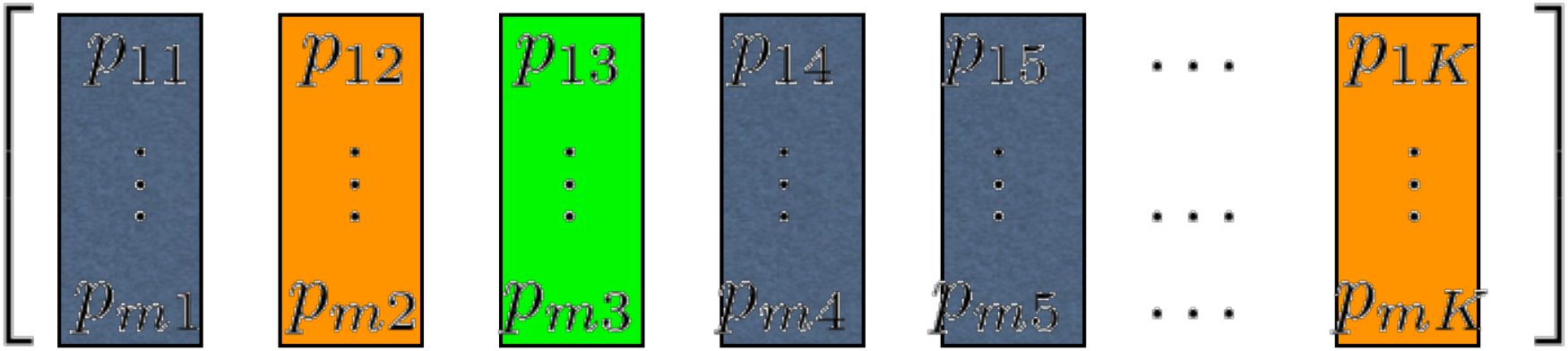
Can we remove the translation component?

No …

Each object has a different centroid, but there is only one centroid per frame.

Motion based Segmentation by Factorization

For N (unsegmented) rigid objects tracked across m frames:



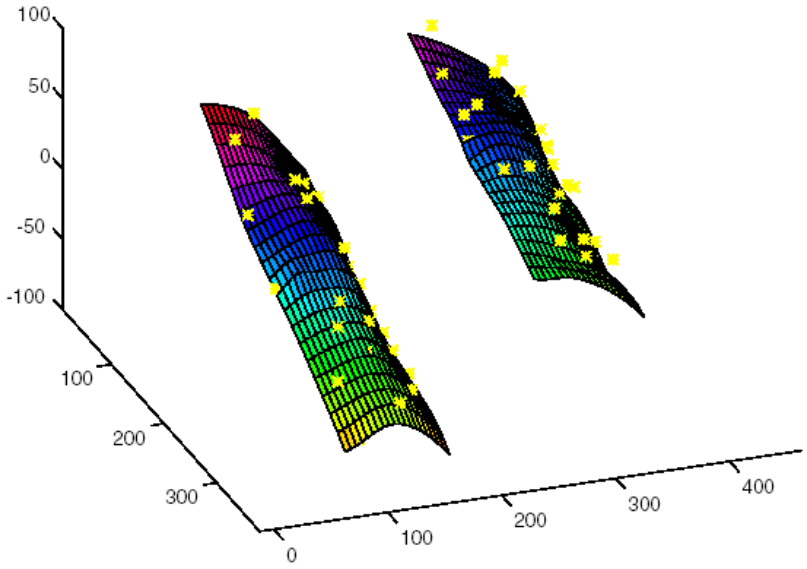
Find SVD of W: $W = UDV^T$

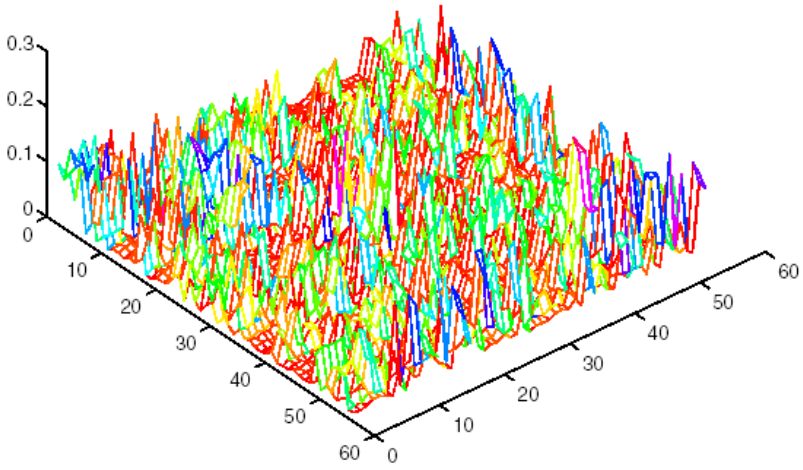
Find "shape interaction matrix":

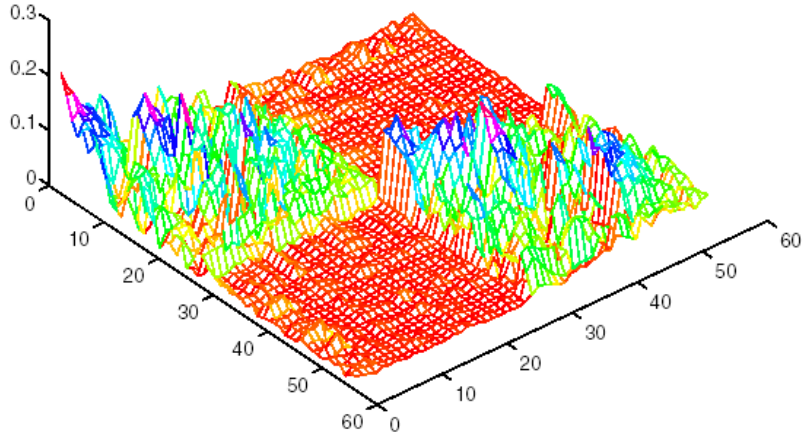
$$Z = VV^T$$

Swap rows and columns of Z until it is block diagonal.

Multiple indep. moving objects







Problems:

Sensitive to noise, outliers, biased tracking …

Cannot handle shared motions modes

Ex:

$$M = \begin{bmatrix} M^1 & M^2 & \dots & M^N \end{bmatrix}$$ With rank 16

How many objects? 16 = 4N ⟹ N = 4

But with shared translation:

$$T^1 = T^2 = \dots = T^N = T$$

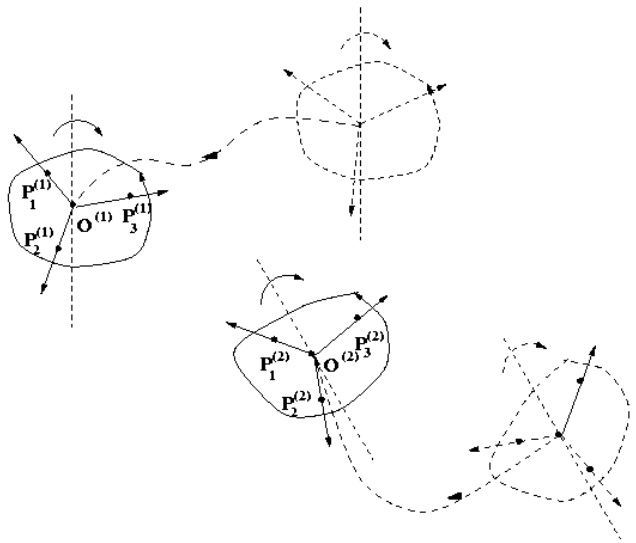
M has at most rank 3N + 1 !!

16 = 3N + 1 ⟹ N = 5

Motion Segmentation from Dynamics

Lublinerman, Sznaier, Camps ‘06

Observations:



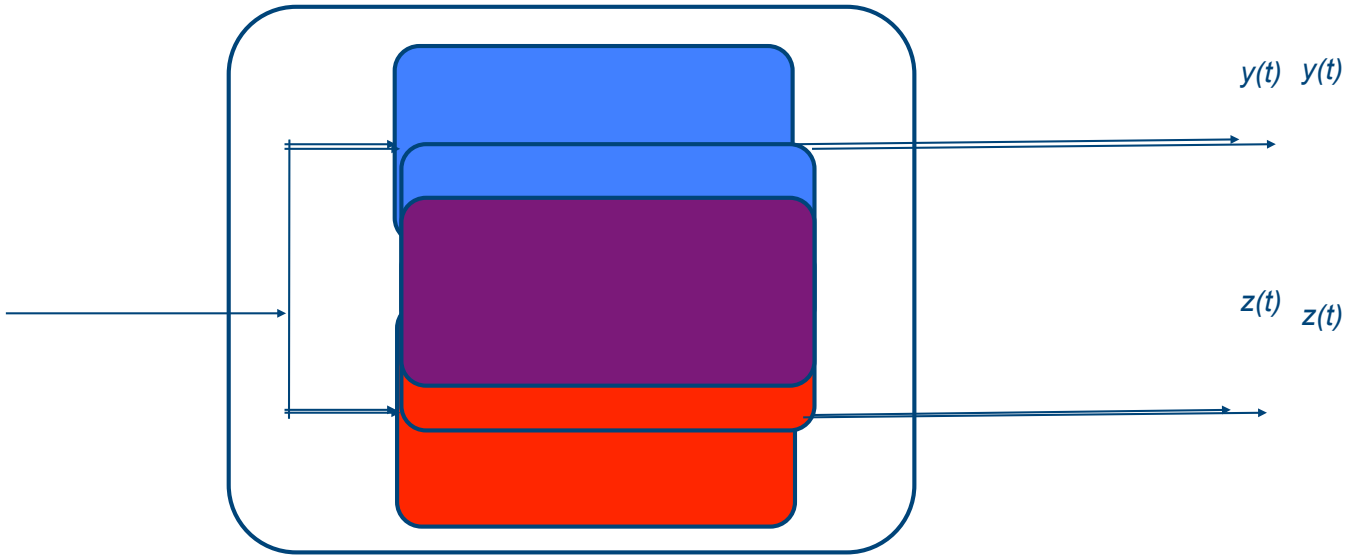
•Temporal information is not used: swapping rows in W does not affect the end result

•The motion of **pairs of points** from a **single object** can be described by the motion of **a** basis attached to the object but does not carry information about the motion of the other objects.

(unobservable states)

•The motion of **pair of points** from **two different objects** can be described by the motion of **two** basis, each of them attached to one of the objects.

Dynamic Data Association



A system explaining independent sequences has higher complexity than a system explaining correlated data.

Look for simplest **joint** models.

(no need to explicitly find the models!)

Main Ideas:

Group points according to the complexity of a linear operator required to explain their spatio-temporal evolution.

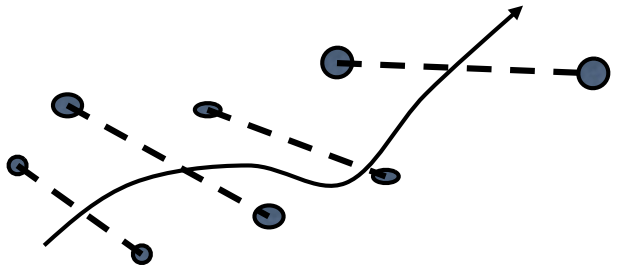
The order of the operator can be estimated by computing the rank of its Hankel matrix.

Dynamics based Segmentation

$$W = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_1 \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix}$$

n points, m frames

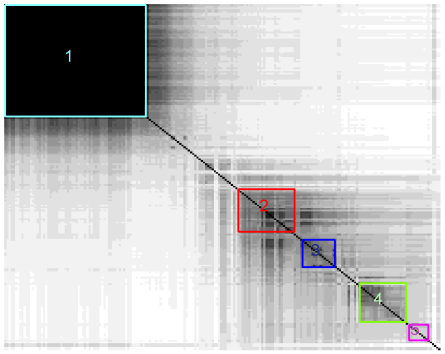
Relative distance between $\mathbf{p}_i$ and $\mathbf{p}_j$:



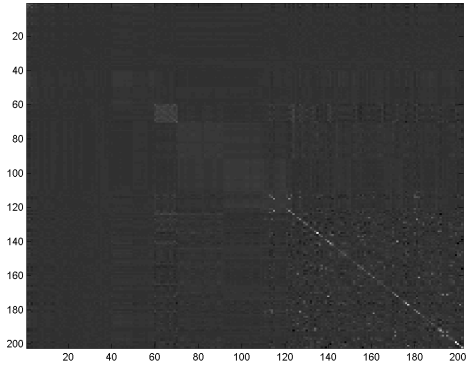
$$d_t^{i,j} = p_{ti} - p_{tj}$$

$$H^{i,j} = \begin{bmatrix} d_1^{i,j} & d_2^{i,j} & \cdots & d_{m/2}^{i,j} \\ d_2^{i,j} & d_3^{i,j} & \cdots & d_{m/2+1}^{i,j} \\ \vdots & \vdots & \vdots & \vdots \\ d_{m/2}^{i,j} & \cdots & \cdots & d_m^{i,j} \end{bmatrix}$$

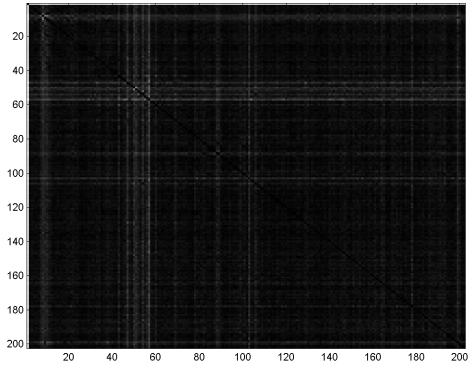
If pi and pj move together, the order of the dynamics needed to explain their motion has lower order than when they move independently.



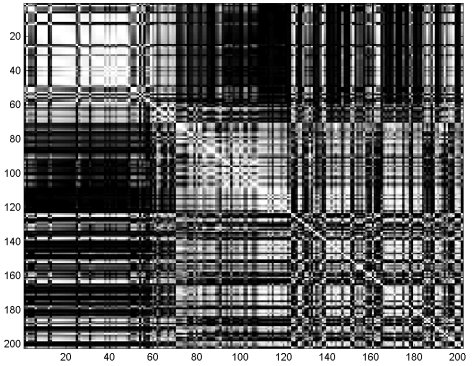
Hankel



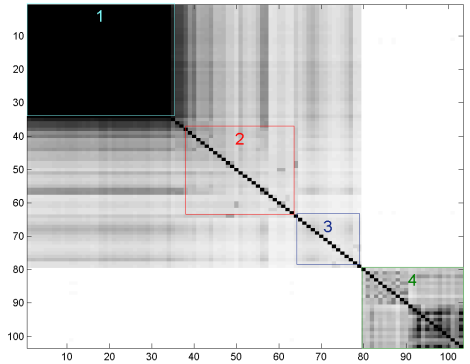
Costeira Kanade



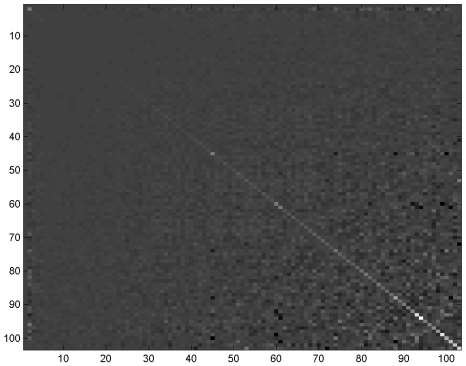
Irani



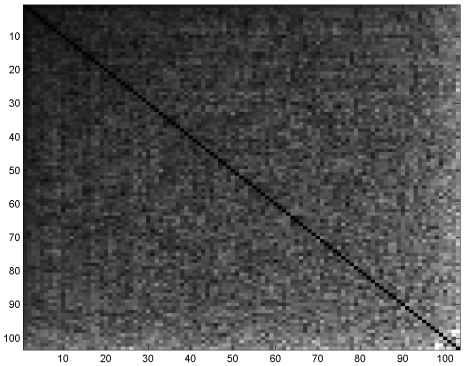
GPCA



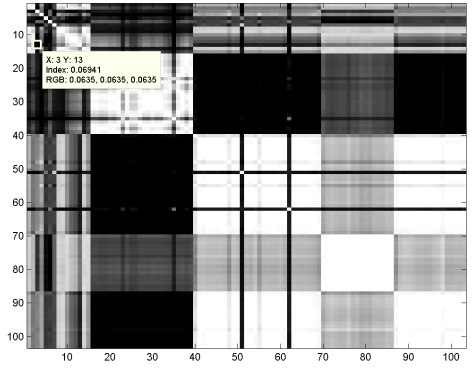
Hankel



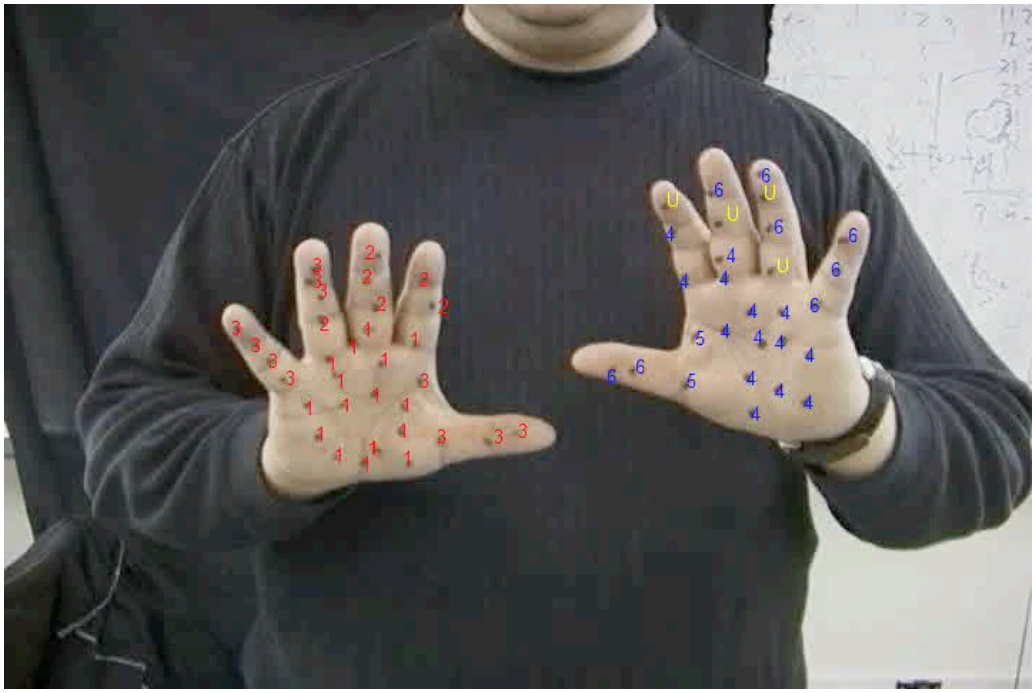
Costeira Kanade

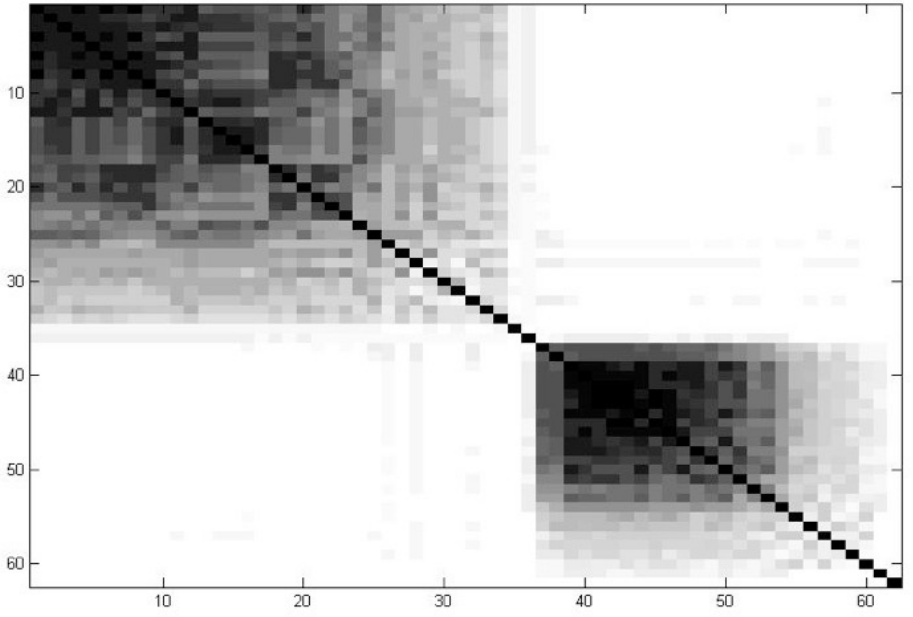


Irani

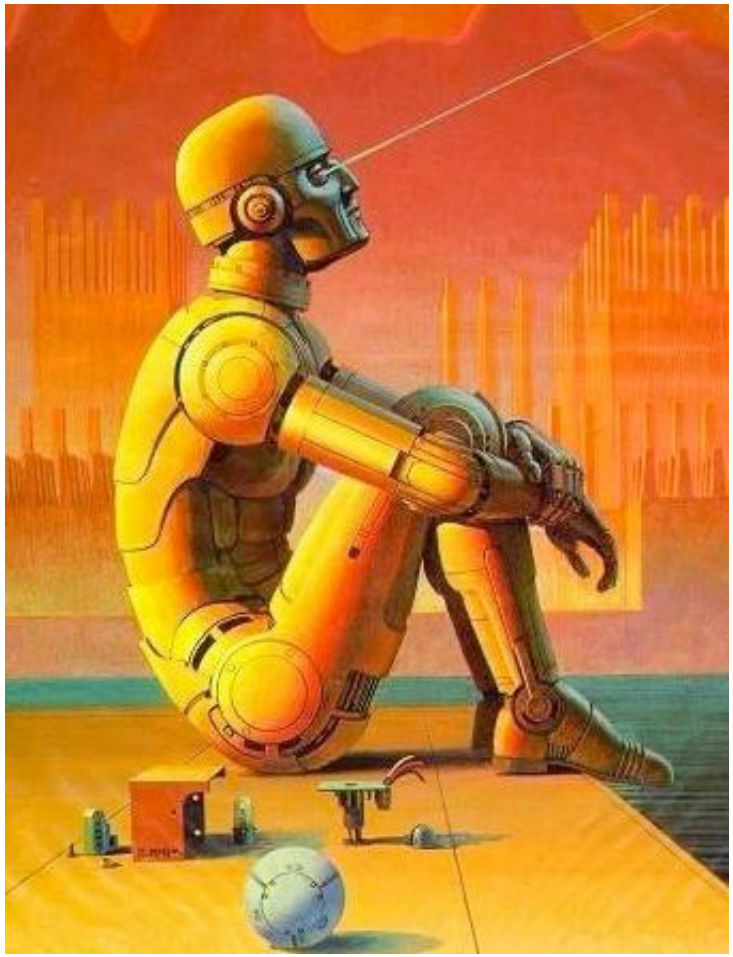


GPCA




10
20
30
40
50
60
10
20
30
40
50
60

3D Object Recognition



Credit: some slides have been adopted from Fergus, Torralba,Fei,Fidler, and Grauman

The Ultimate Goal

Applications:

Industrial inspection, quality control

Surveillance and security

Assisted living

Human-computer interfaces

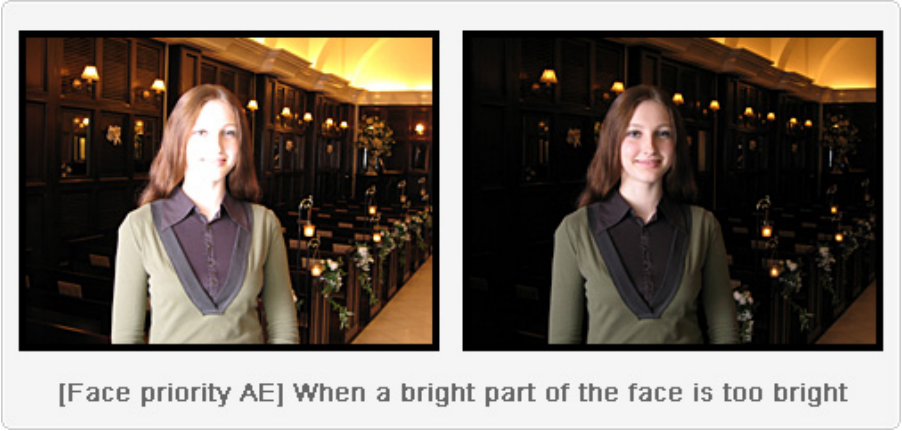
Medical image analysis

Reverse engineering

Image databases

More Applications:

Computational Photography



Improving online search

flickr

Ask Images

picsearch

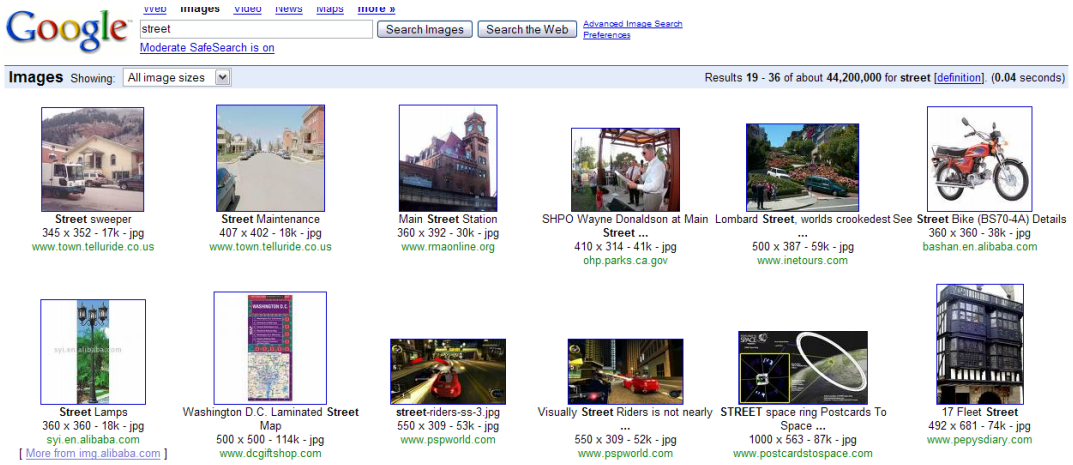
Google Image Search

webshots

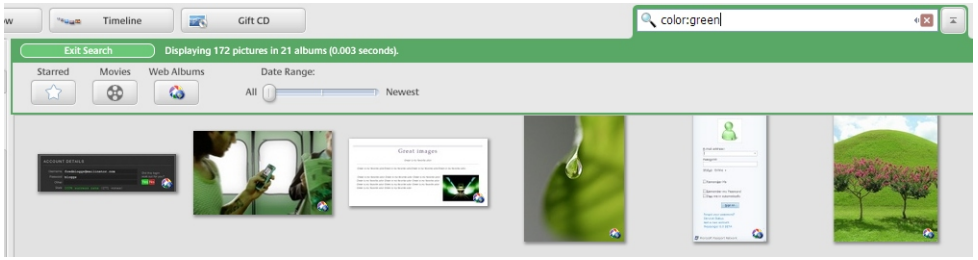
Cydral Image & Site Search

altavista

Query: STREET

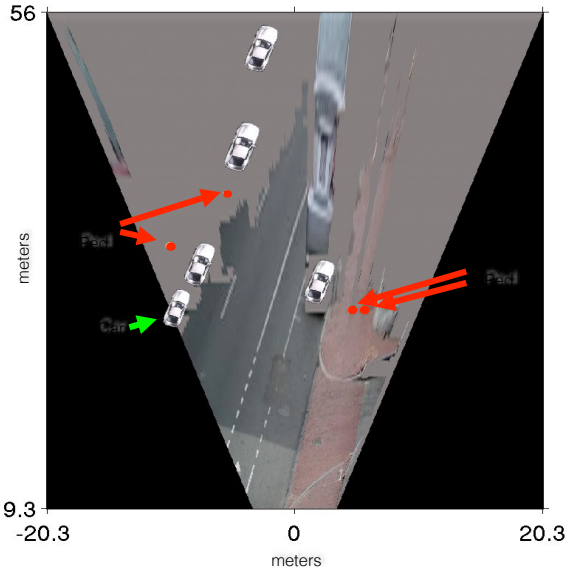


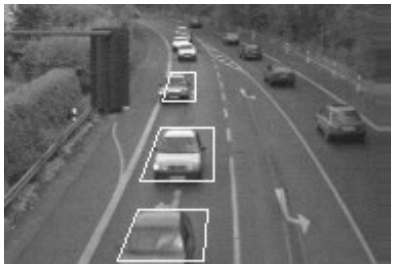
Organizing photo collections



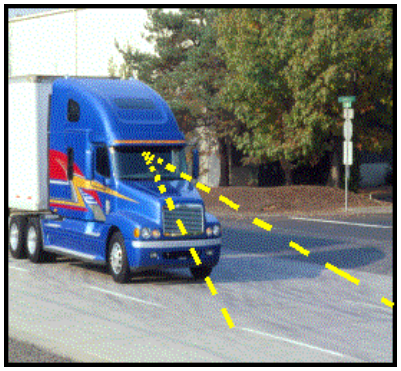
Assisted driving

Pedestrian and car detection





Lane detection



- Collision warning systems with adaptive cruise control
- Lane departure warning systems
- Rear object detection systems

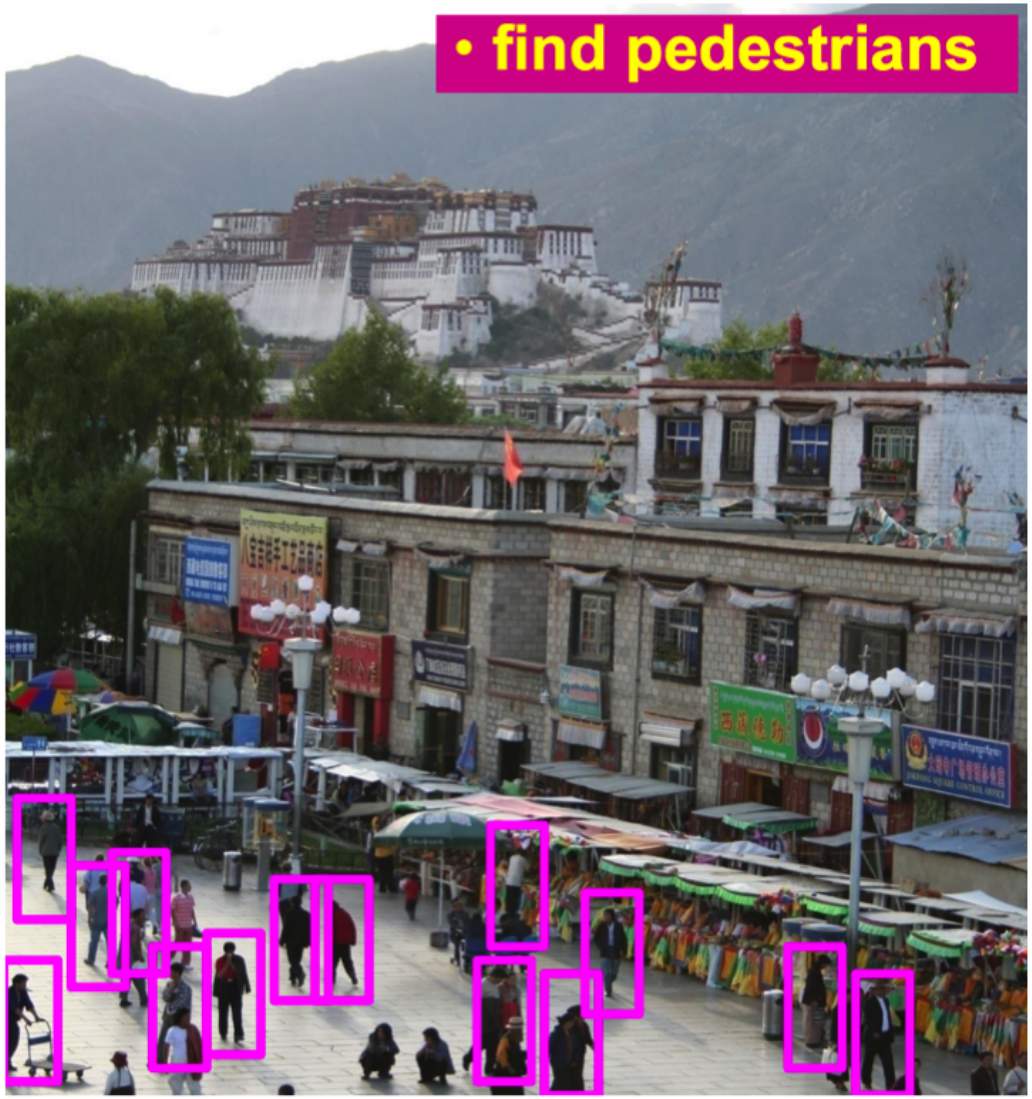
What does Object Recognition Involve?

What does Object Recognition Involve?

Detection:
Are there people?

What does Object Recognition Involve?

Detection:
Where are the people?



What does Object Recognition Involve?

Verification:
Is this a lamp?

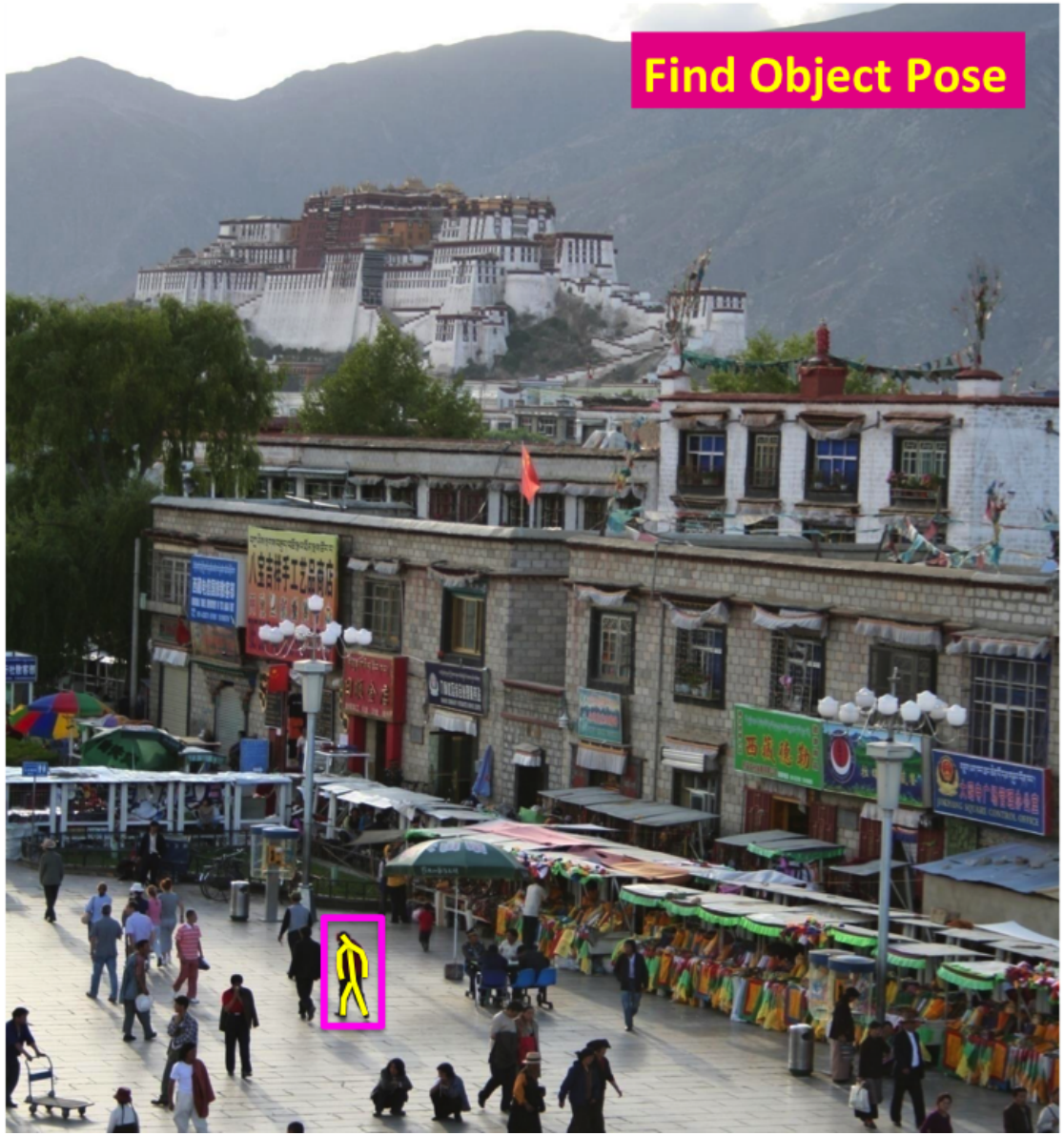
What does Object Recognition Involve?

Identification: Is this Potala Palace?

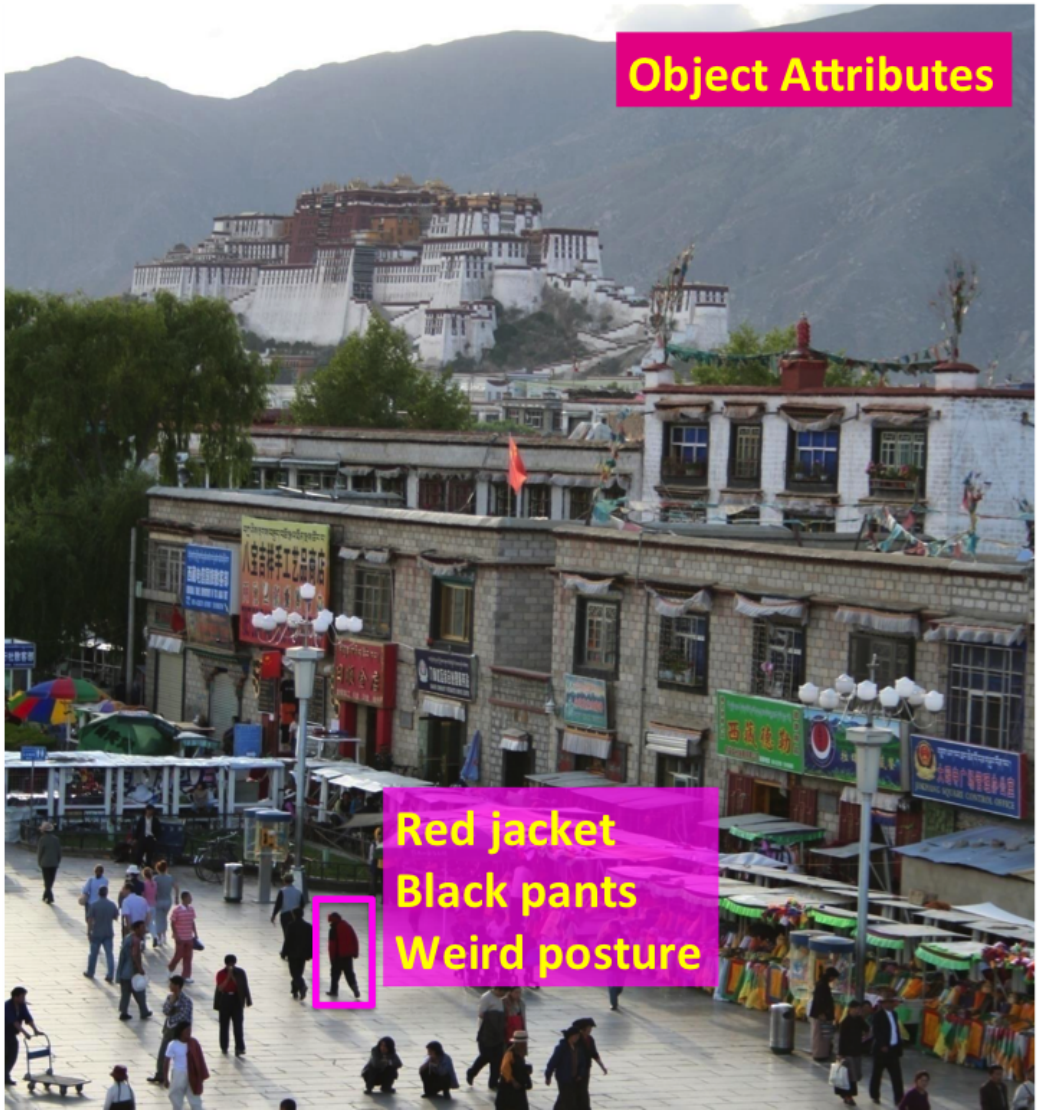
What does Object Recognition Involve?

Object Categorization

What does Object Recognition Involve?

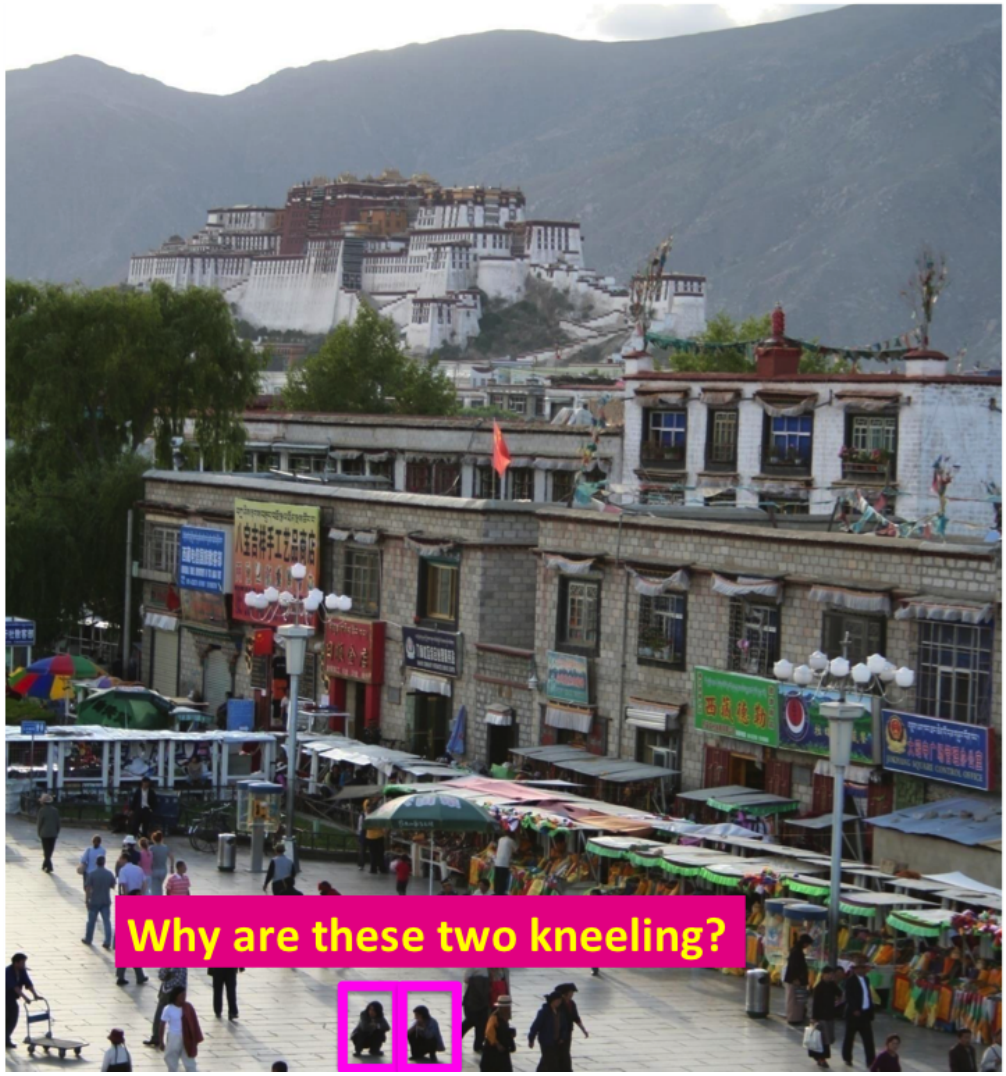


What does Object Recognition Involve?

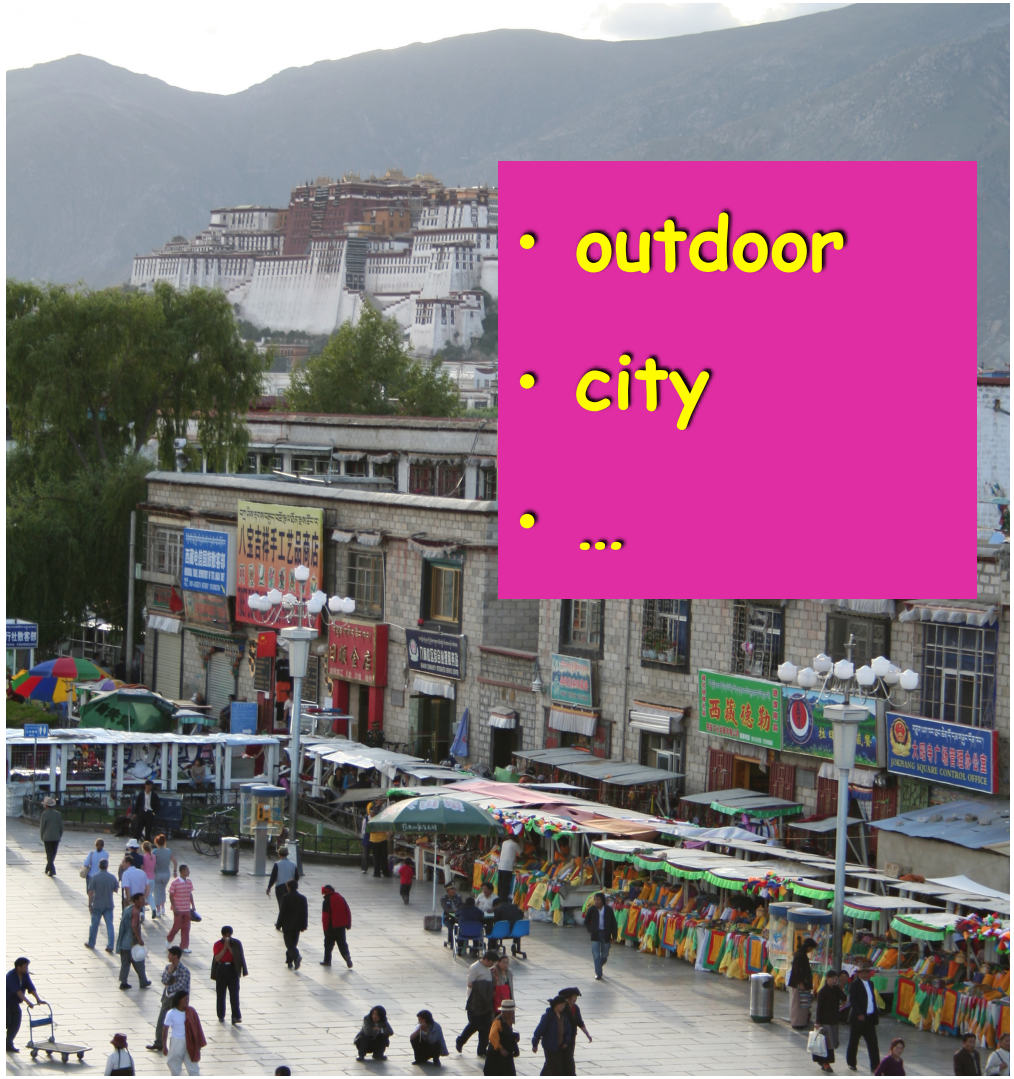


What does Object Recognition Involve?

What does Object Recognition Involve?



What does Object Recognition Involve?



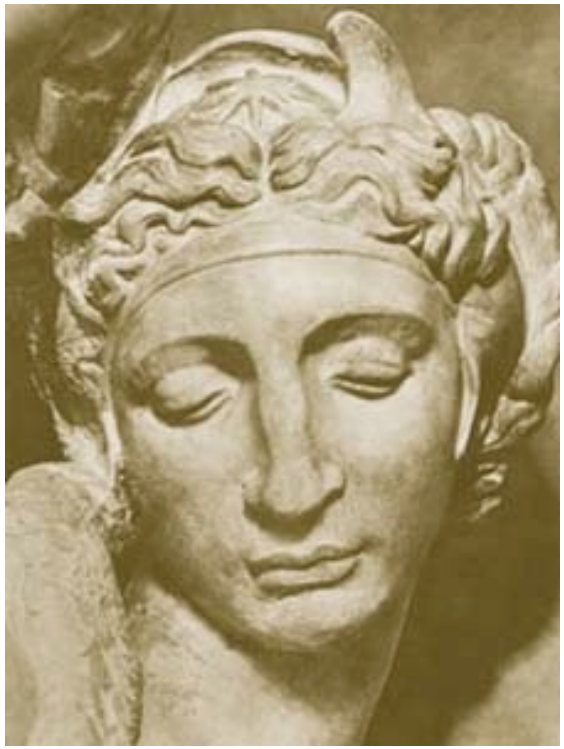
Scene and Context Categorization

How many object categories are there?

Biederman 1987

Not an easy task …

Challenges 1: view point variation



Michelangelo 1475-1564

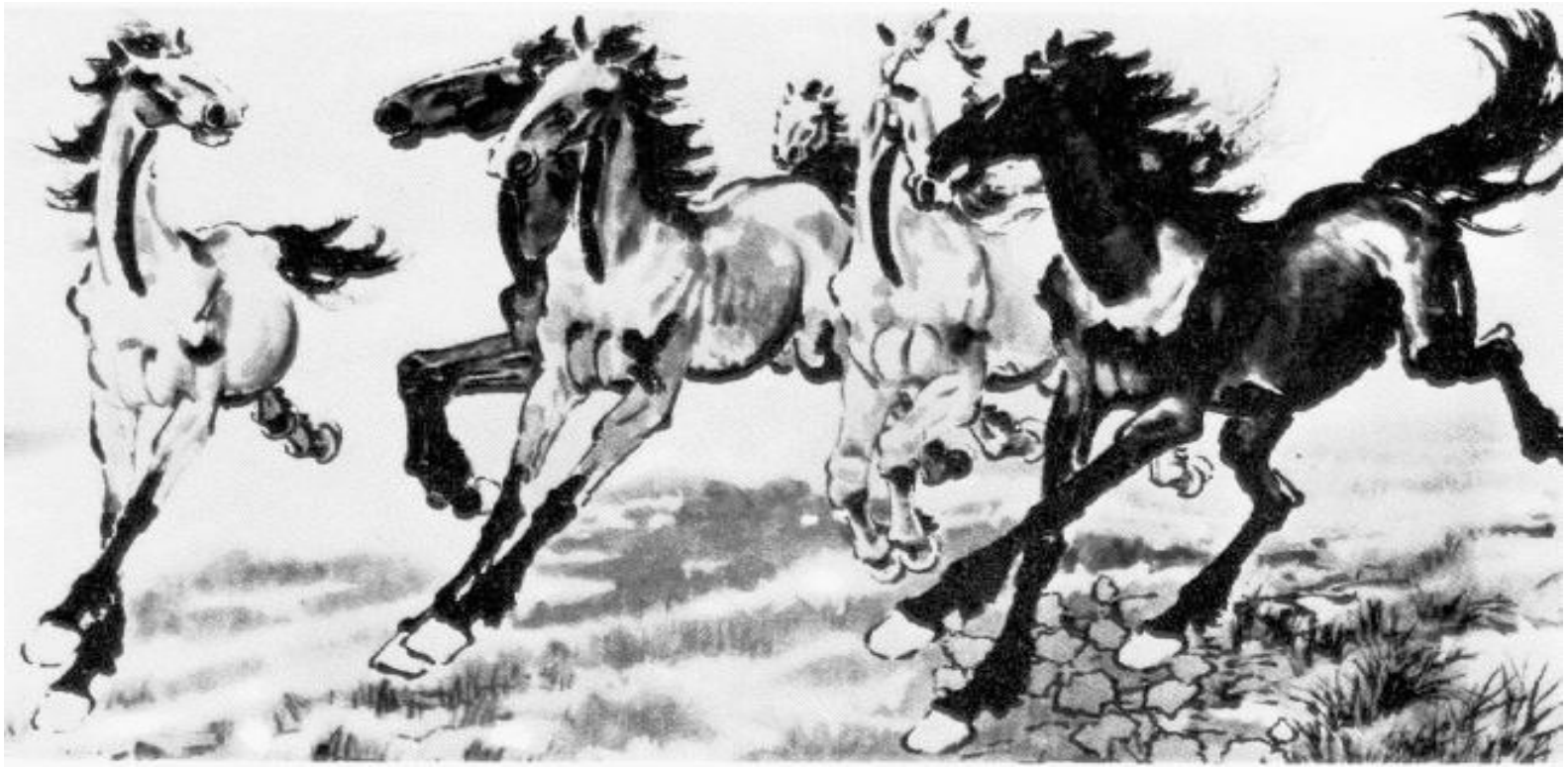
Challenges 2: illumination

Challenges 3: occlusion

Magritte, 1957

Challenges 4: scale

Challenges 5: deformation



Xu, Beihong 1943

Challenges 6: background clutter

Klimt, 1913

History: single object recognition

Challenges 7: intra-class variation







~10,000 to 30,000

History: single object recognition

- Dickinson, 1991
- Camps and Shapiro, 1992
- Sclaroff, 1993
- Nayar, 1995
- Lowe, et al. 1999, 2003
- Mahamud and Herbert, 2000
- Ferrari, Tuytelaars, and Van Gool, 2004
- Rothganger, Lazebnik, and Ponce, 2004
- Moreels and Perona, 2005
- …

History: early object categorization

Turk and Pentland, 1991
Belhumeur, Hespanha, & Kriegman, 1997
Schneiderman & Kanade 2004
Viola and Jones, 2000

- Amit and Geman, 1999
- LeCun et al. 1998
- Belongie and Malik, 2002

- Schneiderman & Kanade, 2004
- Argawal and Roth, 2002
- Poggio et al. 1993

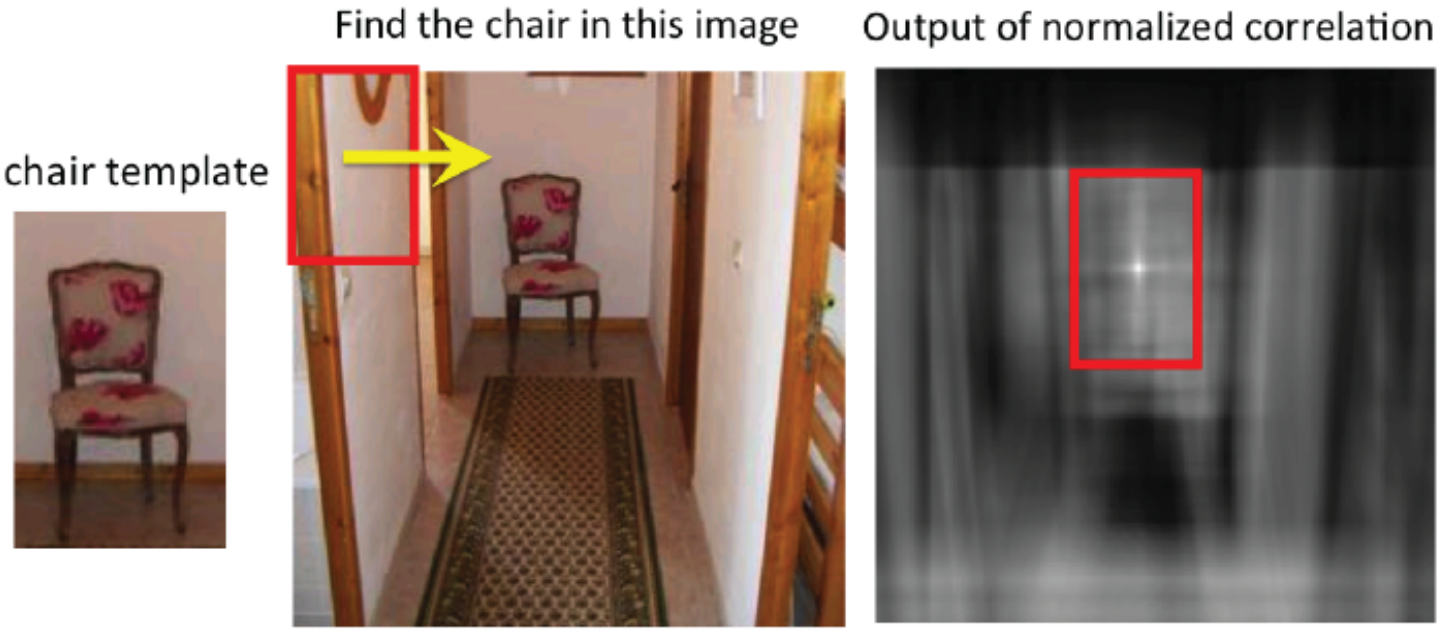
How would we approach this problem?

Let's try using what we already know to do DETECTION

Template Matching

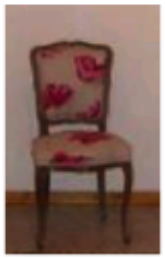
Template Matching

Normalize cross-correlation with a template (filter)



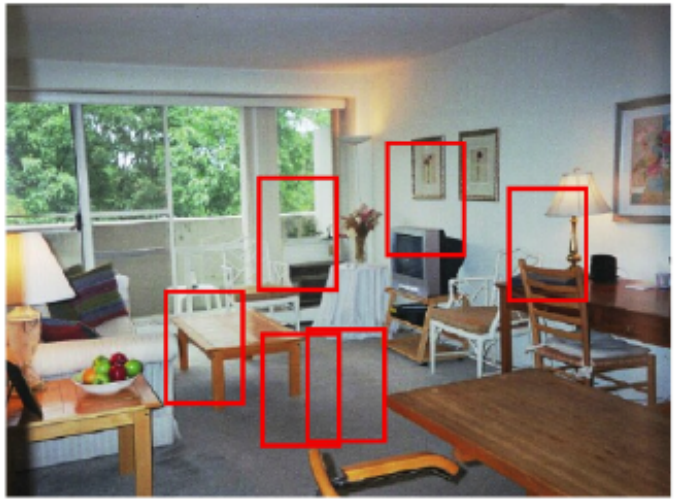
Template Matching

Normalize cross-correlation with a template (filter)

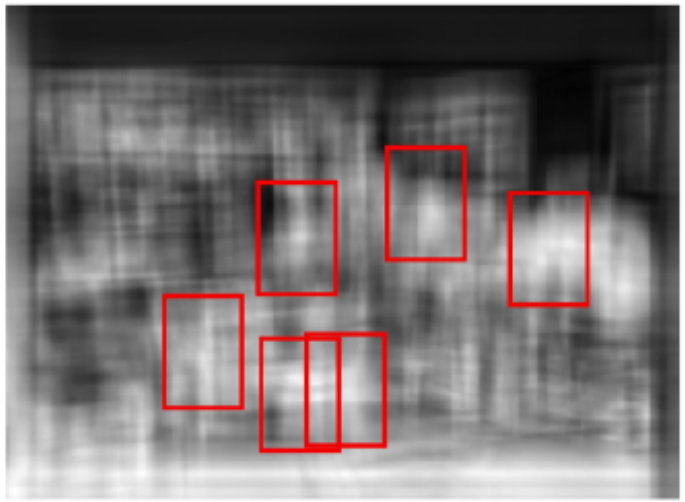


template

Find the chair in this image



Pretty much garbage
Simple template matching is
not going to make it



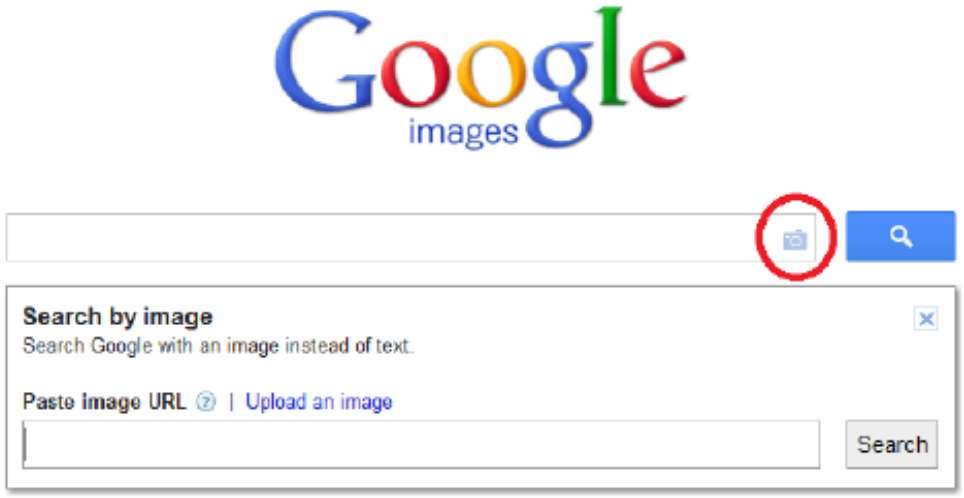
How would we approach this problem?

Let's try using what we already know to do DETECTION

Large sale Retrieval

Store lots of images, recognize one by finding the most similar in the database. Google Approach.

Recognition via Retrieval



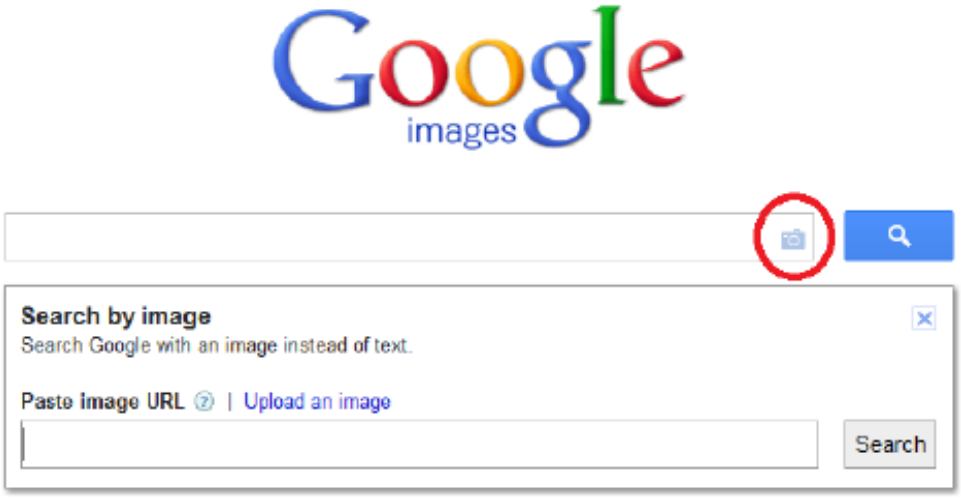
query

Recognition via Retrieval

query

reasonable result!

Recognition via Retrieval



query

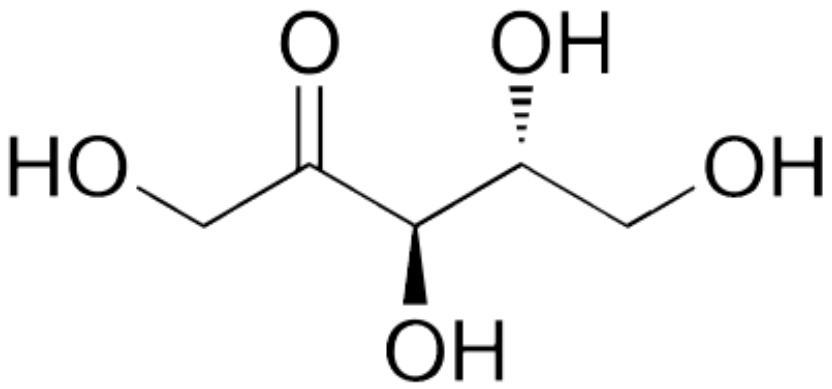
Recognition via Retrieval

query

less reasonable result, but somewhat similar?

Recognition via Retrieval

Recognition via Retrieval



not a reasonable result!

Indexing for Fast Retrieval

PCA: Eigenimages

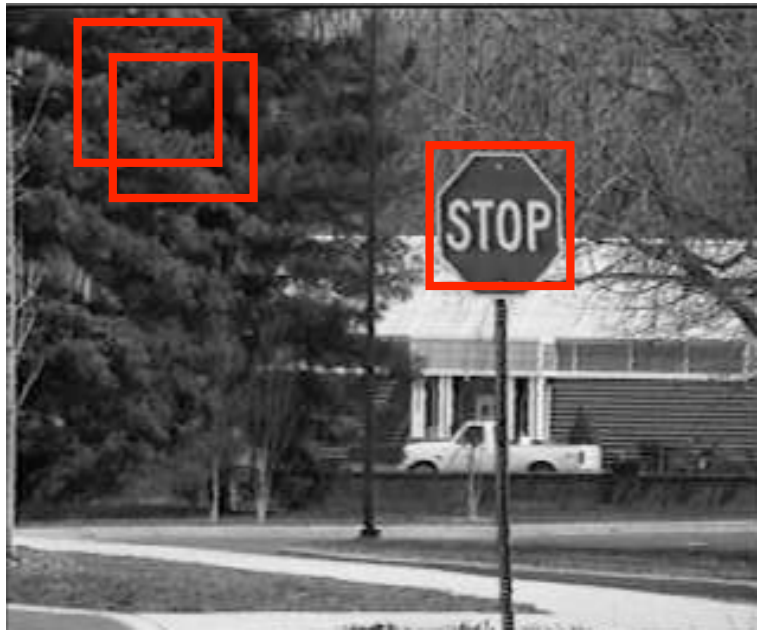
Local Features: Bag of Words

Template Matching

Objects can be represented by storing sample images or "templates"

Stop sign template

Hypotheses fromTemplate Matching



- Place the template at every location on the given image.
- Compare the pixel values in the template with the pixel values in the underlying region of the image.
- If a "good" match is found, announce that the object is present in the image.

- Possible measures are: SSD, SAD, Cross-correlation, Normalized Cross-correlation, max difference, etc.

Limitations of Template Matching

If the object appears scaled, rotated, or skewed on the image, the match will not be good.

Solution:

Search for the template and possible transformations of the template:

Not very efficient! (but doable …)

Limitations of Template Matching

It uses *global* information: it is sensitive to occlusion.

Limitations of Template Matching

It uses pixel values: it is illumination and sensor dependent.

Eigenimages

The appearance of an object in an image depends on several things:

- Viewpoint
- Illumination conditions
- Sensor
- The object itself (ex: human facial expression)

In principle, these variations can be handled by increasing the number of templates.

Eigenimages: Using multiple templates

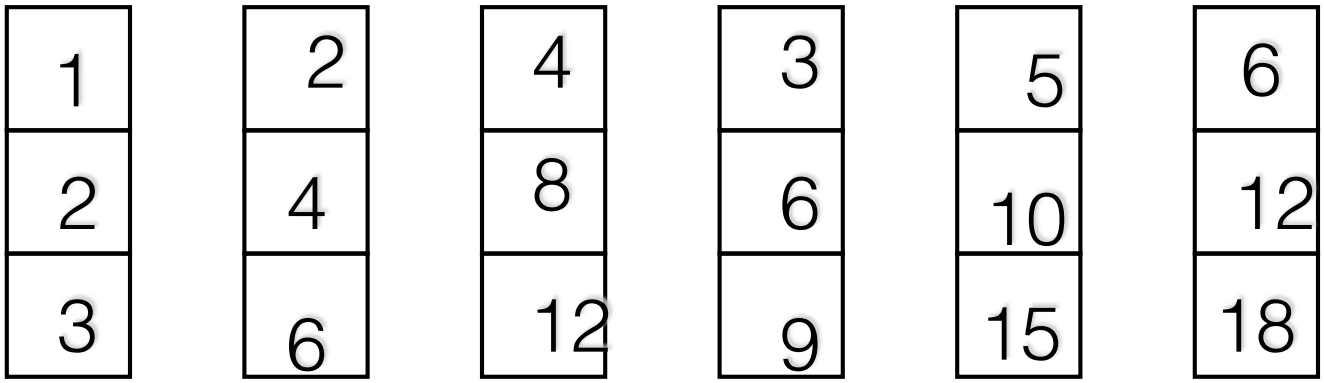
- The number of templates can grow very fast!
- We need:
  - An efficient way to store templates
  - An efficient way to search for matches

- Observation: while each template is different, there exist many similarities between the templates.

Efficient Image Storage

Toy Example: Images with 3 pixels

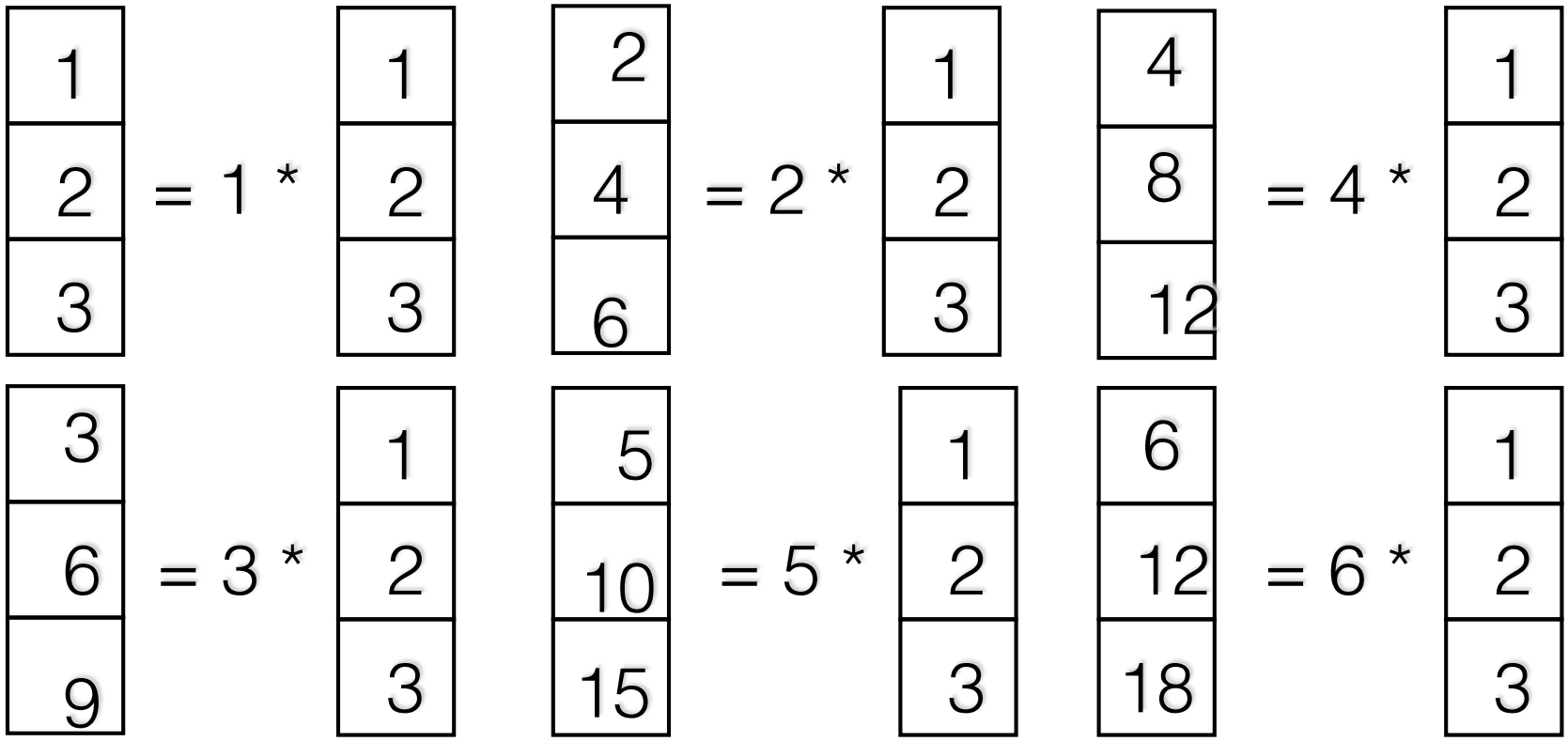
Consider the following 3x1 templates:



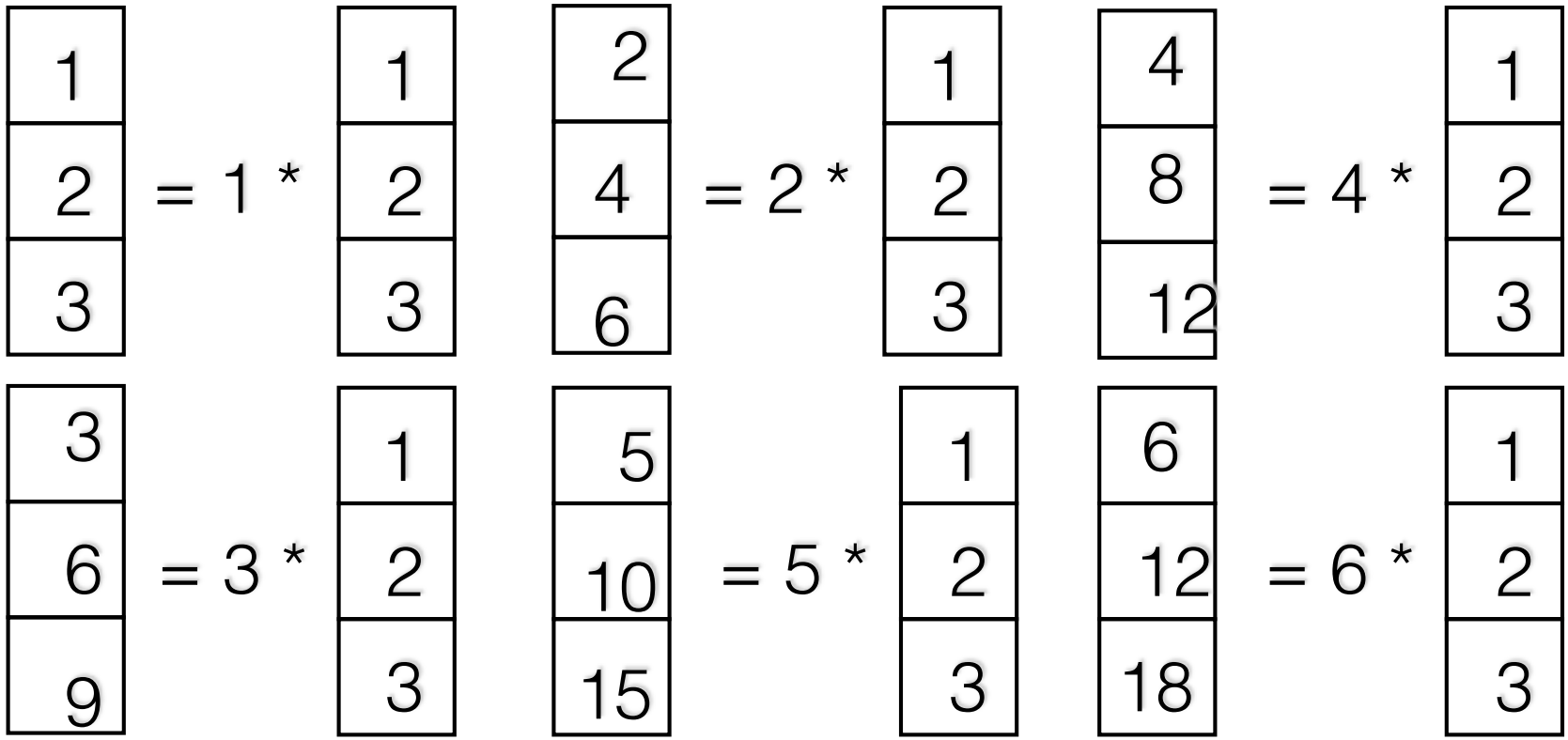
If each pixel is stored in a byte, we need 3 x 6 = 18 bytes

Efficient Image Storage

Looking closer, we can see that all the images are very similar to each other: they are all the same image, scaled by a factor:



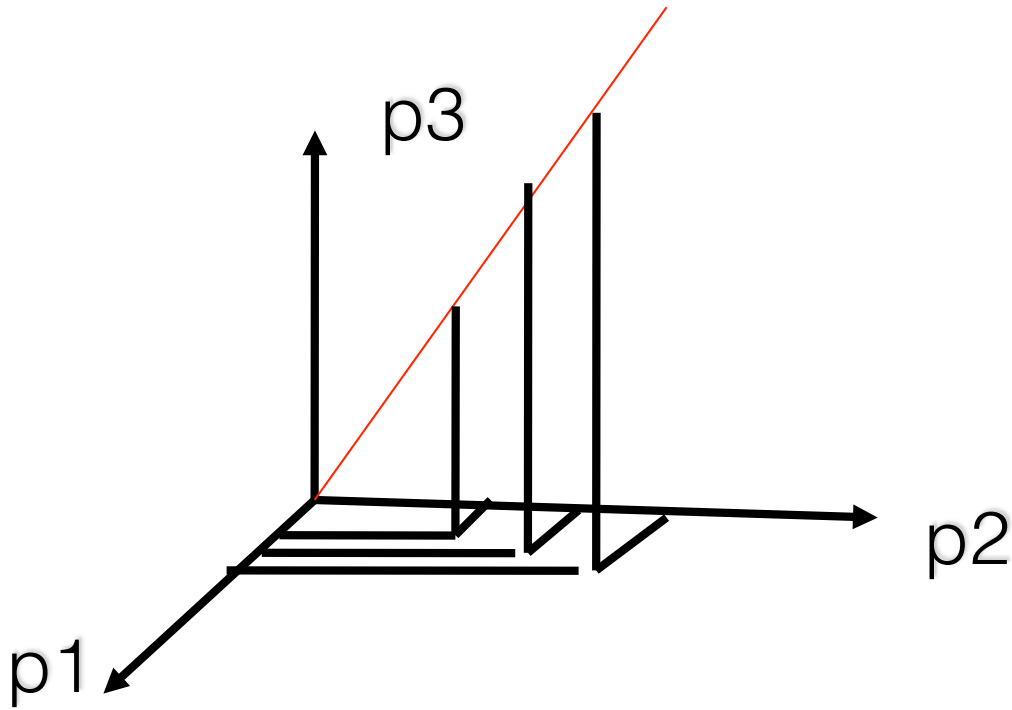
Efficient Image Storage



They can be stored using only 9 bytes (50% savings!):
Store one image (3 bytes) + the multiplying constants (6 bytes)

Geometrical Interpretation:

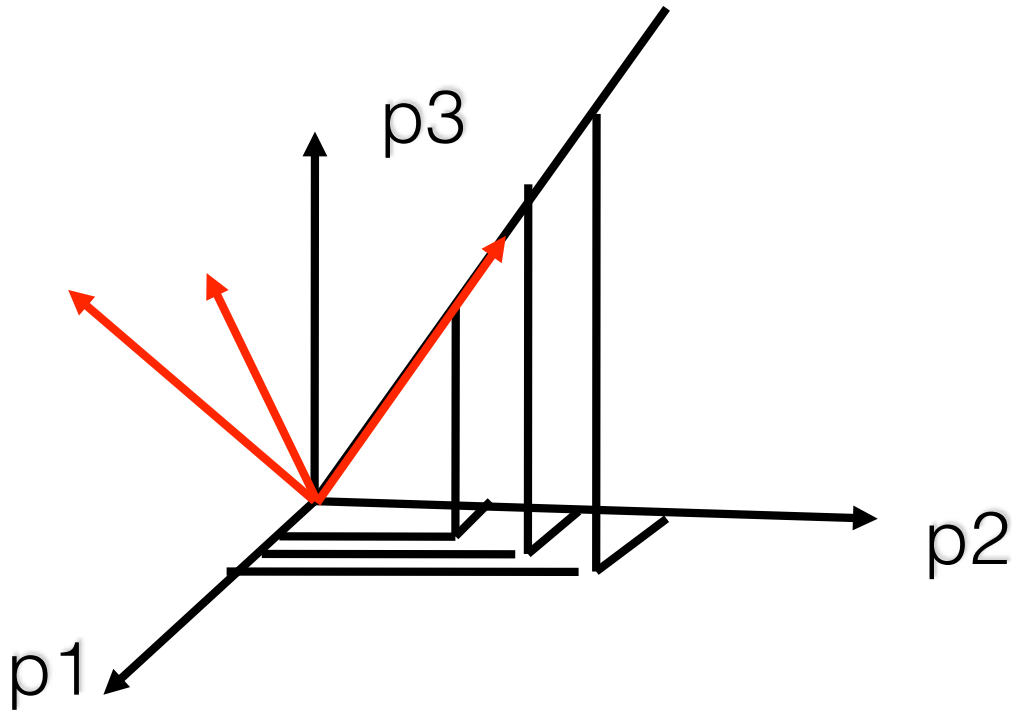
Consider each pixel in the image as a coordinate in a vector space. Then, each 3x1 template can be thought of as a point in a 3D space:



But in this example, all the points happen to belong to a line: a 1D subspace of the original 3D space.

Geometrical Interpretation:

Consider a new coordinate system where one of the axes is along the direction of the line:



In this coordinate system, every image has only one non-zero coordinate: we **only** need to store the direction of the line (a 3 bytes image) and the non-zero coordinate for each of the images (6 bytes).

Principal Component Analysis (PCA)

Given a set of templates, how do we know if they can be compressed like in the previous example?

- The answer is to look into the correlation between the templates
- The tool for doing this is called PCA

PCA Theorem

Let x_1 x_2 … x_n be a set of n N^2 x 1 vectors and let $\bar{x}$ be their average:

$$\mathbf{x}_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN^2} \end{bmatrix} \qquad \bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{i=n} \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN^2} \end{bmatrix}$$

Note: Each N x N image template can be represented as a N^2 x 1 vector whose elements are the template pixel values.

PCA Theorem

Let X be the N^2 x n matrix:

$$X = \left[\begin{array}{cccc} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{array} \right]$$

Note: subtracting the mean is equivalent to translating the coordinate system to the location of the mean.

PCA Theorem

Let Q = X X^T be the N^2 x N^2 matrix:

$$Q = XX^T = \left[\begin{array}{cccc} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{array} \right] \left[\begin{array}{c} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{array} \right]$$

Notes:

1. Q is square
2. Q is symmetric
3. Q is the ***covariance*** matrix
4. Q can be very large (remember that N^2 is the number of pixels in the template)

PCA Theorem

Theorem:

Each x_j can be written as: $\mathbf{x}_j = \bar{\mathbf{x}} + \sum_{i=1}^{i=n} g_{ji}\mathbf{e}_i$

where e_i are the n eigenvectors of Q with non-zero eigenvalues.

Notes:

1. The eigenvectors e_1 e_2 … e_n span an ***eigenspace***
2. e_1 e_2 … e_n are N^2 x 1 orthonormal vectors (N x N images).
3. The scalars g_{ji} are the coordinates of x_j in the space.
4. $$g_{ji} = (\mathbf{x}_j - \bar{\mathbf{x}}).\mathbf{e}_i$$

Using PCA to Compress Data

Expressing x in terms of $e_1 \ldots e_n$ has not changed the size of the data

However, if the templates are highly correlated many of the coordinates of x will be zero or closed to zero.

Using PCA to Compress Data

Sort the eigenvectors e_i according to their eigenvalue:

$$\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$$

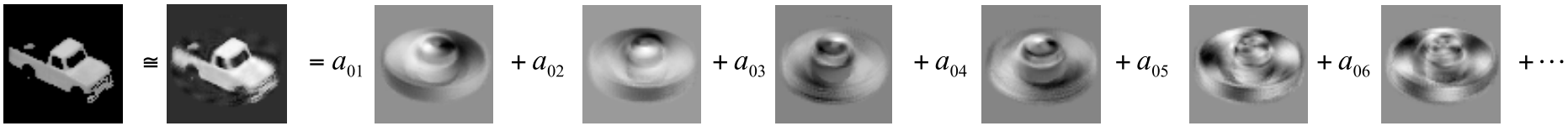
- Assuming that $\lambda_i \approx 0 \text{ if } i > k$

- Then

$$\mathbf{x}_j \approx \bar{\mathbf{x}} + \sum_{i=1}^{i=k} g_{ji}\mathbf{e}_i$$

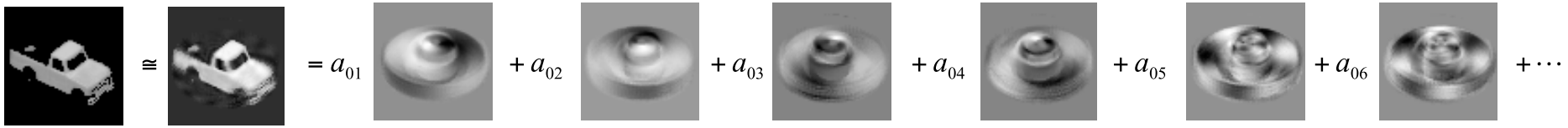
Eigenspaces: Efficient Image Storage

- Use PCA to compress the data:
  - each image is stored as a k-dimensional vector
  - Need to store k N x N eigenvectors
- $k << n << N^2$



Eigenspaces: Efficient Image Comparison

- Use the same procedure to compress the given image to a k-dimensional vector.
- Compare the compressed vectors:
  - Dot product of k-dimensional vectors
  - $k << n << N^2$



Implementing PCA

Need to find "first" k eigenvectors of Q:

$$Q = XX^T = \left[\begin{array}{cccc} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{array} \right] \left[\begin{array}{c} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{array} \right]$$

Q is N^2 x N^2 where N^2 is the number of pixels in each image.
For a 256 x 256 image, N^2 = 65536 !!

Finding ev of Q

$Q=XX^T$ is very large. Instead, consider the matrix $P=X^TX$

- Q and P are both symmetric, but $Q \neq P^T$
- Q is N^2 x N^2, P is n x n
- n is the number of training images, typically $n << N$

Finding ev of Q

Let e be an eigenvector of P with eigenvalue λ:

$$Pe = \lambda e$$
$$X^T X e = \lambda e$$
$$X X^T X e = \lambda X e$$
$$Q(Xe) = \lambda(Xe)$$

Xe is an eigenvector of Q also with eigenvalue λ!

Singular Value Decomposition (SVD)

Any m x n matrix X can be written as the product of 3 matrices:

$$X = UDV^T$$

Where:

- U is m x m and its columns are orthonormal vectors
- V is n x n and its columns are orthonormal vectors
- D is m x n diagonal and its diagonal elements are called the singular values of X, and are such that:

$$\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n \geq 0$$

SVD Properties

$$X = U D V^T$$

- The columns of U are the eigenvectors of XX^T
- The columns of V are the eigenvectors of X^TX
- The squares of the diagonal elements of D are the eigenvalues of XX^T and X^TX

Algorithm EIGENSPACE_LEARN

Assumptions:

1. Each image contains one object only.
2. Objects are imaged by a fixed camera .
3. Images are normalized in size N x N:

   The image frame is the minimum rectangle enclosing the object.
4. Energy of pixels values is normalized to 1:

   $\sum_i \sum_j I(i,j)^2 = 1$
5. The object is completely visible and unoccluded in all images.

Algorithm EIGENSPACE_LEARN

Getting the data:

For each object o to be represented, o = 1, …,O

1. Place o on a turntable, acquire a set of n images by rotating the table in increments of $360^o/n$
2. For each image p, p = 1, …, n:
   1. Segment o from the background
   2. Normalize the image size and energy
   3. Arrange the pixels as vectors $\mathbf{x}^o_p$

Algorithm EIGENSPACE_LEARN

Storing the data:

1. Find the average image vector $\bar{\mathbf{x}} = \frac{1}{n.o} \sum_{o=1}^{O} \sum_{p=1}^{n} x_p^o$

2. Assemble the matrix X:

$$X = \left[\begin{array}{cccc} \mathbf{x}_1^1 - \bar{\mathbf{x}} & \mathbf{x}_2^1 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n^o - \bar{\mathbf{x}} \end{array} \right]$$

3. Find the first k eigenvectors of XX^T: $e_1,\ldots,e_k$
(use X^TX or SVD)

4. For each object o, each image p:

- Compute the corresponding k-dimensional point:

$$\mathbf{g}_p^o = \left[\begin{array}{cccc} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_k \end{array} \right] (\mathbf{x}_p^o - \bar{\mathbf{x}})$$

Algorithm EIGENSPACE_IDENTIF

Recognizing an object from the DB:

1. Given an image, segment the object from the background

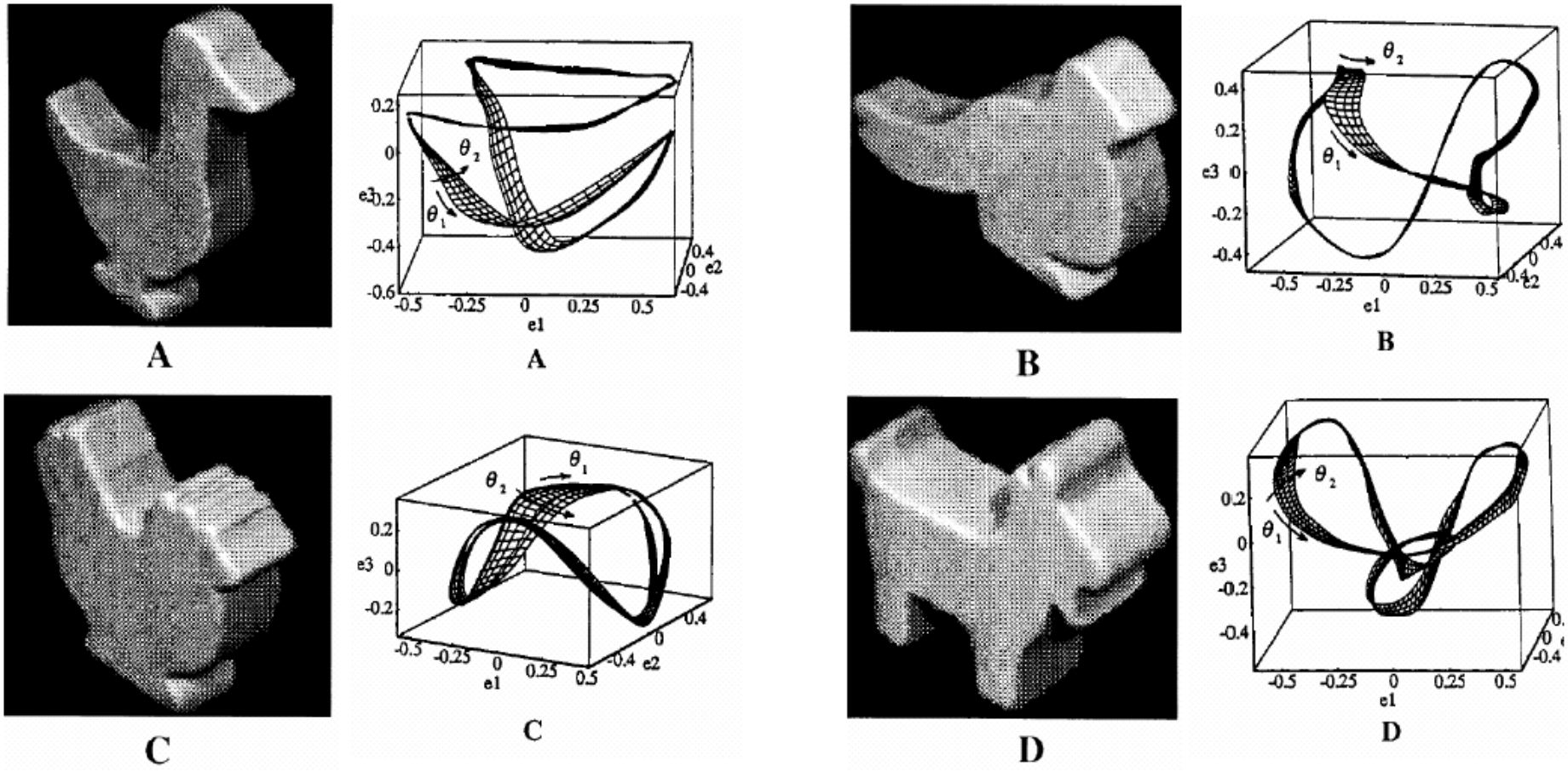
2. Normalize the size an energy, write it as a vector **i**

3. Compute the corresponding k-dimensional point:

$$\mathbf{g} = \left[\begin{array}{cccc} \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_k \end{array} \right] (\mathbf{i} - \bar{\mathbf{x}})$$

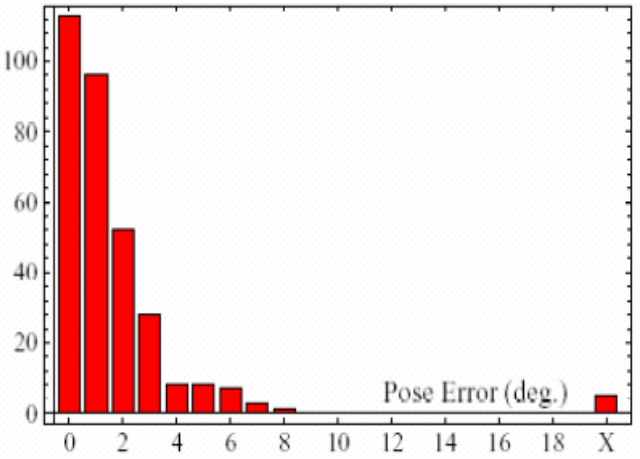
4. Find the closest $\boldsymbol{g}^{o}_{p}$ k-dimensional point to ***g***

Appearance Manifolds



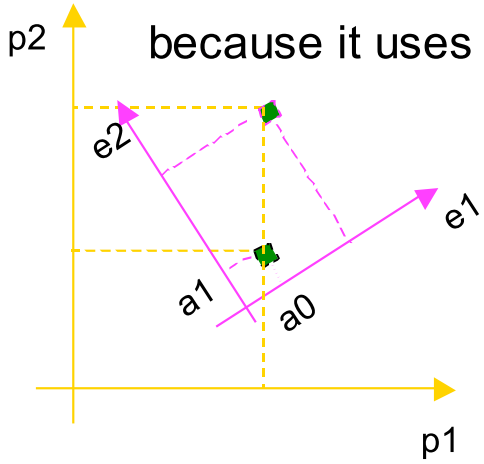
[Murase & Nayar, 1996]

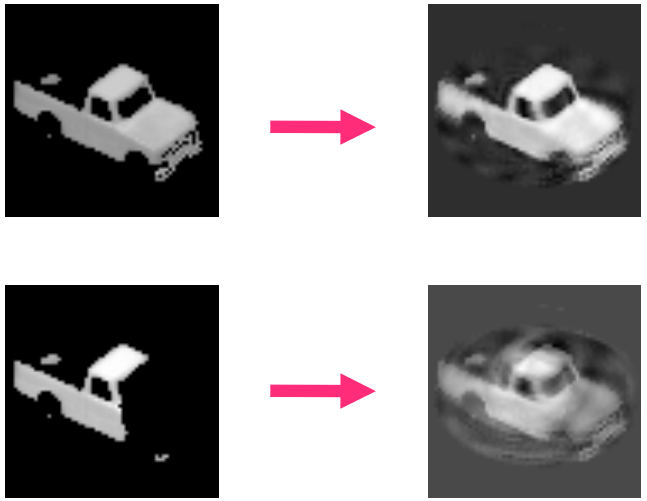
Appearance Manifolds



Murase & Nayar

PCA has problems with occlusion





Example 3: ABPs and ABRs

Use local appearance:

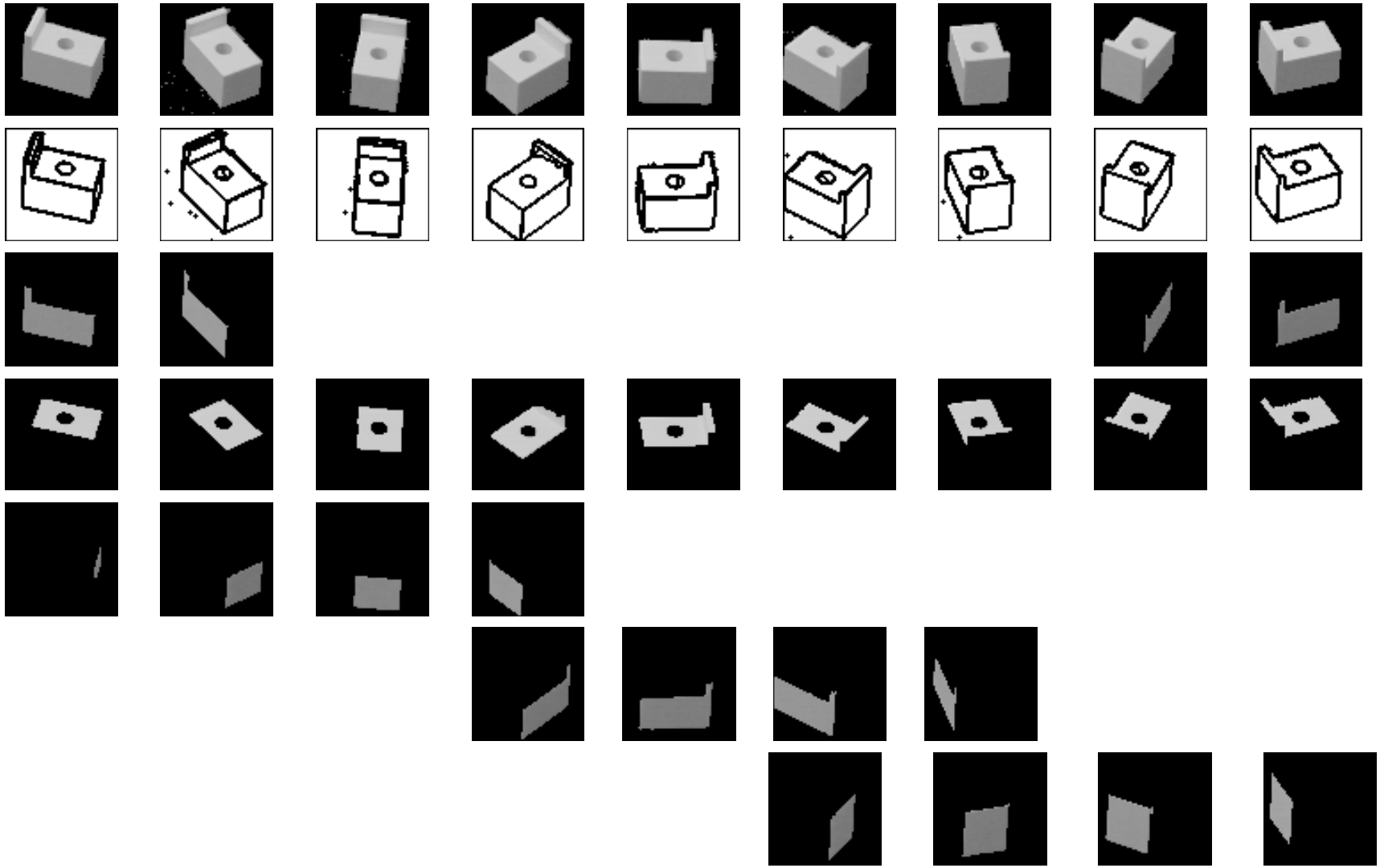
PARTS and RELATIONS

- Can model general-form objects
- Can deal with occlusion and clutter
- Can handle translation and scaling
- It is robust to segmentation problems

Parts from Images

PARTS are image regions segmented using some segmentation algorithm.

Appearance-Based Parts



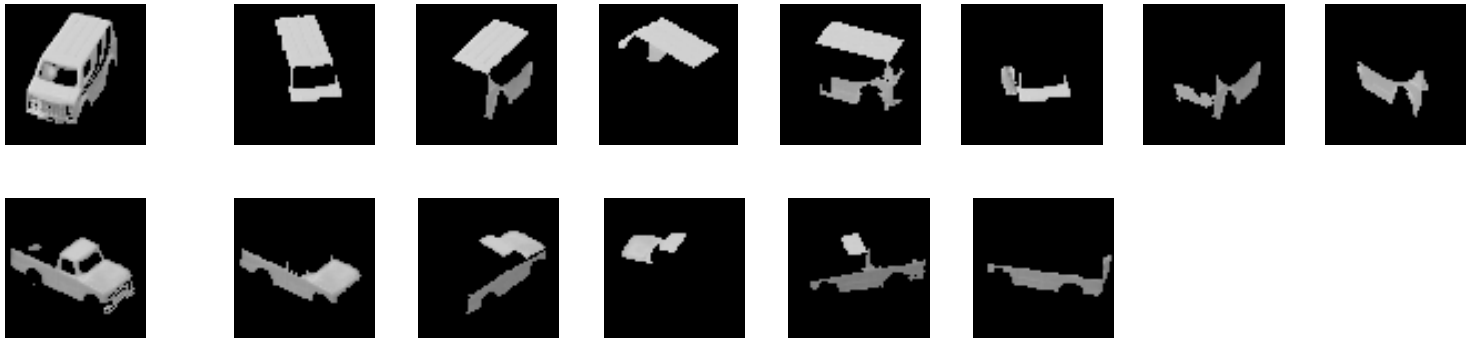
Appearance-Based Relations

Several objects can have similar parts

Use spatial relations to discriminate

Represent relations using PCA

Examples of ABRs



Recognition in cluttered scenes

Segment the given image

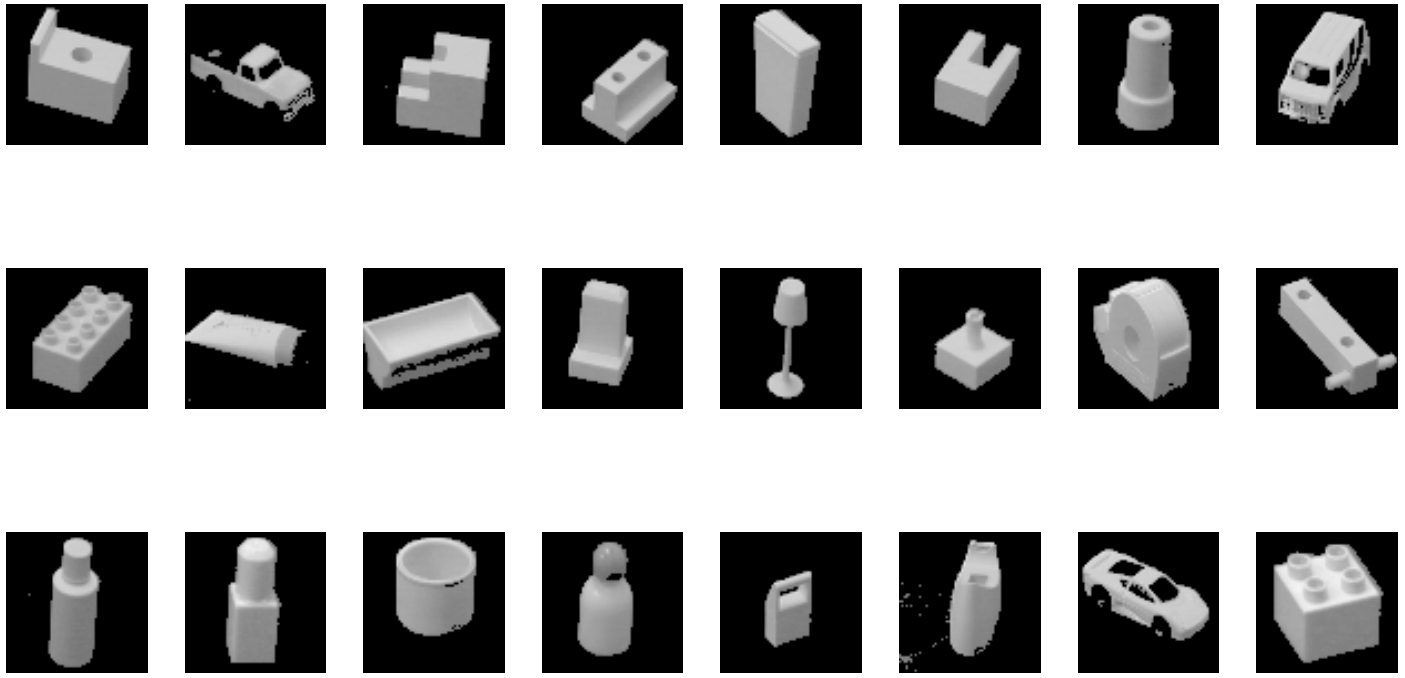
PARTS:

- Project regions into the ABP eigenspace
- Make hypotheses if $d < T1$

RELATIONS:

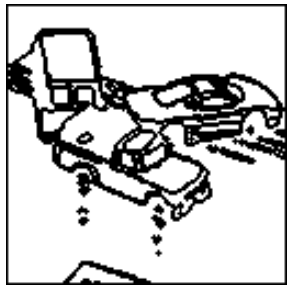
- Project adjacent regions with $T1 < d < T2$ into the ABR eigenspace
- Make hypotheses if $d' < T3$

Object Database

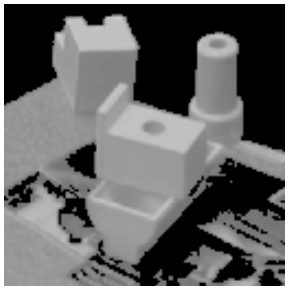
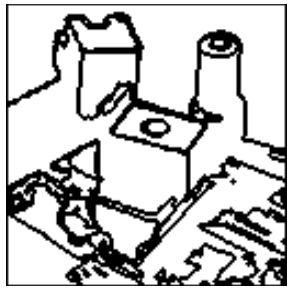


24 objects; 110 ABPs (69 shape groups/3 levels); 130 ABRs

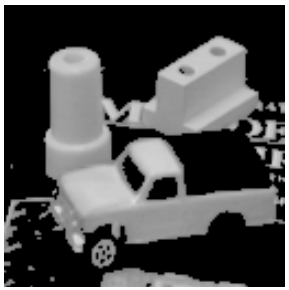
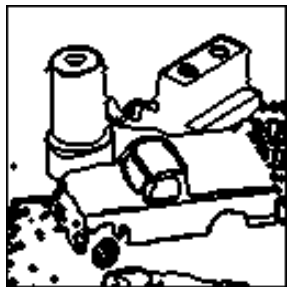
Some Examples

Identity	Score	Pose
Sport Car	0.61	302.77
Truck	0.19	4.29
Ambulance	0.10	11.52

Identity	Score	Pose
Ccube	1.36	41.12
Stamp	0.89	0.64
Holecube	0.89	14.47
Sink	0.68	152.94

Identity	Score	Pose
Stamp	1.91	93.66
Twohole	0.96	181.63
Truck	0.4	93.67